

INTRODUCTION and PERIPHERAL PROCESSOR

Training Manual

CONTROL DATA INSTITUTE

6000 SERIES INTRODUCTION AND PERIPHERAL PROCESSORS
TRAINING MANUAL

SECOND EDITION

FOR TRAINING PURPOSES ONLY

This manual was compiled and written
by instructional personnel of

CONTROL DATA INSTITUTE
CONTROL DATA CORPORATION

Publication Number 60250400
November, 1968

This manual obsoletes the 6000 Training Supplement, publication number 60241500 (previously 011568). Also, Chapter 1 of 6600 Central Processor Control and Memory, Volume I, publication number 020167 has been deleted from that book.

Copyright 1968 by Control Data Corporation
Printed in the United States of America

FOREWORD

This manual is intended to serve as an introductory text to 6000 Series Computer Systems and to provide logical analysis of the peripheral processor units of this series. As an introductory text, this manual discusses common design philosophies of the CONTROL DATA® 6400, 6500, and 6600 Computer Systems. During the course of study, the Peripheral Processor Customer Engineering Diagrams Manual and the Chassis I Wire Tabs should be available as additional reference material.

It is assumed that the reader of this manual has a thorough knowledge of number systems, Boolean algebra, and basic computer concepts.

In any technical writing effort, possibilities of errors are always present. Although Control Data Institute makes a conscious effort to minimize errors in its publications, errors are nevertheless inevitable. If you would like to make the existence of errors known, or would like to make comments or suggestions concerning the manual, you might find the Comments Sheet at the end of the manual to be of help. Forward your comments to the Educational Development Department of Control Data Institute, 3101 East 80th Street, Minneapolis, Minnesota 55440.

®Registered trademark of Control Data Corporation

CONTENTS

CHAPTER 1. INTRODUCTION TO 6000 SERIES COMPUTER SYSTEMS. 1
Introduction 1
Market Area 1
Transistor 1
New Concepts 2
Peripheral and Control Processors 3
PPU Data Channels 4
PPU Physical Characteristics 5
Central Processor 6
Central Processor Operation 6
Operating Registers 6
Instruction Format 7
Exchange Jump 8
Physical Concepts 9
Logic Circuits 9
Cooling 11
Magnetic Core Storage 11
Signal Transmission 11
6000 Series Components 12
6400 System 12
6600 System 12
6400 or 6600 System 12
6612 Display Console 12
6603 Disk File 12
6638 Disk File System 14
6622 Magnetic Tape Controller 15
626-B Magnetic Tape Transport 15
6681 Data Channel Converter 16

CHAPTER 2. 6000 SERIES LOGIC ANALYSIS 17
Circuits 17
Packaging 17

Cooling 17
Basic Circuit Analysis 18
AND and OR Combinations 18
Logic Diagram Symbols 18
Flip-Flop with Clear/Set Input 21
Circuit Loading 21
Line Drivers 22
Twisted-Pair Driver 22
Coaxial Cable Driver 23

CHAPTER 3. PERIPHERAL AND CONTROL PROCESSORS 25
Introduction 25
Peripheral and Control Processor Characteristics 27
Timing 27
PPU Timing 27
Registers 28
A Register 29
P Register 30
Q Register 30
K Register 30
Slot 31
A Adder 31
P Incrementor 34
Q Register 34
General Theory of Adders 34
Q Adder 37
K Register 39
Input/Output 41
I/O Instructions and Signals 41
Master Clear 41
Disconnect (75) 41
Function (76 or 77) 41
Activate (74) 42
Input (70, 71) 42
Output (72, 73) 42
Communication with Central Memory and Central Processor 43
6400 Only Central Program Monitor 43
Central Busy 44
Exchange Jump 44
Central Read 44
Central Write 44
Dead Start 45
Load 45
Sweep 46
Dump 46
6400 Peripheral Processors 46
26XX Instructions 46
27XX Instructions 46
Additional X Register 46
Module Changes 46

CHAPTER 4. CONCEPT OF THE 6600 CENTRAL PROCESSOR 47
Introduction 47
Memory Bank Phasing 47
Instruction Stack 47
Summary 49

Block Diagram Analysis 49
Central Memory Address Control 49
Central Memory 53
Instruction Control 54
Instruction Issue 54
Issue Control 55
Reservation Control (Scoreboard) 57
First-Order Conflicts 58
Second-Order Conflicts 60
Third-Order Conflicts 61
Register Exit/Entry Control 61
Entry Control 61
Exit Control 63

CHAPTER 5. THE 6400 COMPUTER SYSTEM 77
Introduction 77
Market Position 77
6400 System General Block Diagram 77
System Characteristics 77
Data Channels 78
Peripheral and Control Processors (PPU's) 78
Central Memory 78
Central Processor Unit (CPU) 78
System Elements 78
System Block Diagram 78
Basic 6400 Computer System 78
Special Purpose Central Processor 78
6681 Data Channel Converter 79
General Timing Comments 79
Central Processor (CPU) 79
Unified Central Processor 79
Operating Registers 79
Central Processor Overview 80
Input 80
Instruction Controls 80
Operating and Control Registers 80
Large Arithmetic Section (LAS) 80
Small Arithmetic Section (SAS) 80
Output 80
Central Processor Operation 80
Input 80
Output 80
Instruction Controls 81
Types of RNI Operation 81
Large Arithmetic Section (LAS) 81
Small Arithmetic Section (SAS) 82
6400 Central Memory Controls (CMC) 82

CHAPTER 6. 6000 SERIES MAGNETIC CORE STORAGE ANALYSES 89
Introduction 89
Peripheral Processor Memory 89
Central Memory (CM) 89
Extended Core Storage (ECS) 89
Detailed Discussion of Core Storage Systems 90
Storage Modules 90
Magnetic Core Memory 90

General Principles 90
Description of Magnetic Cores 91
Storage Module Construction 92
Memory Plane 92
Stack 92
Drive Deck 92
Inhibit Deck 92
Connectors 92
Mechanical Parts 92
Address Selection 92
Drive Circuits 93
Current Source 94
Diverter Circuits 95
Inhibit Circuits 97
Peripheral Processor Memory 99
Storage Sequence Control (SSC) 100
Memory Cycle Path 101
Power 102

CHAPTER 7. 6000 SERIES FLOATING POINT 103
Introduction 103
Packing 104
Unpacking 105
Range Definitions 105
Single/Double Precision 106
Nonstandard Floating-Point Arithmetic 109

CHAPTER 8. REFRIGERATION SYSTEM 111
Introduction 111
Principles of Refrigeration 111
System Description 112
Physical Description 113
Operation 113
Element Description 113
Warning System 115
Preventive Maintenance 116
Daily 116
Weekly 116
General Refrigeration Service 116
Pump-Down Procedure 116
Filter-Drier Replacement 118
Adding Refrigerant 118
Purging 118
Condensing Unit Replacement 118
Chassis Pressure Regulator Valve Servicing 119
Expansion Valve Service 119

APPENDIX A. PERIPHERAL PROCESSOR COMMAND TIMING A-1

FIGURES

Figure	Title	Page
1	Ratio of I/O Time to Actual Computational Time	2
2	6000 Computer System General Block Diagram	3
3	Sequence of Concurrent Operation of Two Jobs	3
4	PPU Block Diagram	4
5	Central Processor Block Diagram	6
6	Instruction Formats	7
7	Symbolic Coding	7
8	Exchange Jump Package	8
9	Program Distribution in Central Memory	8
10	6000 Logic Module	9
11	Back of Chassis Wiring	10
12	Typical 6000 Chassis	10
13	Refrigeration Tubes	10
14	Refrigeration Unit	11
15	6000 Memory Module	11
16	Data Trunk Between Central Memory and Operating Registers	11
17	6612 Display Console	13
18	6603 Disk File	13
19	807 Disk File	14
20	626-B Magnetic Tape Transport	15
21	6681 Data Channel Converter	16
22	Basic Inverter	18
23	Combining Signals	19
24	AND Combination	19
25	OR Combination	20
26	Basic Flip-Flop with Clear/Set Input	20
27	Flip-Flop Logical Representation and Timing Chart	21
28	Base Resistor Loading	22
29	Twisted-Pair Driver	23
30	Coaxial-Cable Driver	24
31	Peripheral Processors	26
32	12- and 24-Bit Instruction Formats	27
33	Four-Phase Clock Regeneration System	27
34	Clock Regeneration Circuit	28
35	Barrel	29
36	(P) → G Register → S Registers	30
37	Elements of the Slot	31
38	A Adder	32
39	Shift Network	33
40	Shift	34
41	Shift Path Control	34
42	P Incrementor	34
43	Q Adder	35
44	Adder Elements	36
45	Carry Network	36
46	Equivalence Circuit	37
47	Carry Input Circuit for Group 0	38
48	Carry-Up	39
49	Carry-Up	39
50	Summation Network	39
51	Q Adder	40
52	K In Slot	40
53	Channel Controls	41
54	Channel Lines	41

FIGURES (Cont)

Figure	Title	Page
55	Input Operation	42
56	Output Operation	43
57	Central P Address to A	43
58	(A) to Central Memory	43
59	Central Busy	44
60	Read Pyramid	44
61	Write Pyramid	45
62	Detailed Timing for Dead Start Pulse	45
63	Instruction Stack	48
64	Timing for Accept Signal Return to Stunt Box	50
65	Tag Bit Positions	52
66	Read/Write Distributor	53
67	Instruction Word Parcels	54
68	Parcel Counter	55
69	Skip Generation	56
70	No Branch Condition	57
71	Scoreboard Block Diagram	59
72	Register Exit/Entry Control Block Diagram	62
73	A General and Overall 6600 Computer Block Diagram	64
74	Central Processor Chassis Layout 131K	65
75	Central Processor Block Diagram	66
76	Central Processor Stunt Box Block Diagram	67
77	6600 Stunt Box Detail Block Diagram	68
78	Hopper	69
79	Scoreboard	70
80	Central Processor Block Diagram	71
81	Shift Functional Unit Block Diagram 1	72
82	Boolean Functional Unit	72
83	Add Functional Unit	73
84	Long Add Unit Block Diagram	73
85	Multiply Functional Unit	74
86	Divide Functional Unit	75
87	Increment Functional Units Block Diagram	76
88	Branch Functional Unit	76
89	Instruction Position Variations	81
90	60-Bit Instruction Word Grouped in Parcels	81
91	6400 System General Block Diagram	84
92	6500 System General Block Diagram	84
93	6400/6500 Chassis Layout (131K)	85
93a	6400 CPU Sequence Control Diagram	86
94	6400 Central Processor Block Diagram	87
95	Central Memory Block Diagram	88
96	Core Selection	90
97	Hysteresis Loop	91
98	Sense Voltage	91
99	Drive Line Connections	93
100	Memory Cycle Timing	94
101	Drive and Inhibit Timing	95
102	Primary Read/Write Current Sources	96
103	Opposing Current Relationship of Inhibits and Write Drive Signals	97
104	Group 1 Inhibit Lines	98
105	Inhibit Transformer Drive Line Sequence	99
106	PPU Memory	100
107	Storage Sequence Control	100

FIGURES (Cont)

Figure	Title	Page
108	Stages of Storage Sequence Control	101
109	Reset Circuit	101
110	Sense Merge	102
111	400 CPS Power Supplies	102
112	Power Supply Usage	102
113	6000 Series Floating-Point Format	103
114	6000 Floating-Point Exponent Chart	108
115	Simple Refrigeration Scheme	112
116	Control Data 6600 Conventional Cooling System	114
117	Simplified Diagram of 6000 Cooling System	115
118	6000 Refrigeration System (New Style)	117

TABLES

Table	Title	Page
1	Peripheral and Control Processor Instruction Times	5
2	Extended Core Storage Transfer Rates for Each Core Storage Module	12
3	Circuit Symbols Used in Logic Diagrams	19
4	Component Variations for Circuit Loading	22
5	Component Variations	22
6	Adder Progression of Two Test Operands	40
7	I/O Channel Lines	41
8	Cycle Times for Functional Units	48
9	Sequential Computer Instruction Execution Time	49
10	6600 Computer Instruction Execution Time	49
11	Hopper Tag Definitions	52
12	Maximum In-Stack Loop	56
13	Functional Unit Codes	60
14	Second Instruction Word Penalty Time	79
15	Tag Sequences	83
16	Signed Coefficient Rules	104
17	Description of Bits 2^{58} and 2^{59}	104
18	Operands for Nonstandard Floating-Point Arithmetic	109
19	Refrigerant 12 Characteristics	112
20	Troubleshooting Chart for 6600 Cooling System	121

CHAPTER 1

INTRODUCTION TO 6000 SERIES COMPUTER SYSTEMS

INTRODUCTION

Before embarking on a study of the 6000 Series Computers, a historical look at their initial development will help in explaining many of the new concepts they employ.

MARKET AREA

Soon after Control Data successfully had marketed its first large-scale digital computer, the 1604, it decided to develop a super-scale computer to fill a specific area of the computer market. This particular area was scientific in nature, and its most indicative group was the Atomic Energy Commission (AEC). The scientific problem solving that these groups did involved the solving of a large and dynamic mathematical model. This called for a very large and very fast central computer.

Several attempts were made to build a machine for this type of problem. However, none of these machines was very successful due to technological problems.

TRANSISTOR

All of these previous attempts were solid state machines using the presently available germanium transistor. Since this most basic component was operating near its maximum expected speed, it seemed that the design problem was unsolvable. However, Control Data Corporation decided to attempt a design that could efficiently handle the type of computer-solving problem the AEC had.

NEW CONCEPTS

Since no more speed could be expected from the transistor, brute force speed of computing was not possible. Instead, conceptual changes in the basic computer system were necessary. Using this principle as a guide line, Control Data started the development of the 6600 Computer.

For the next year or so, many new concepts were developed that would increase the speed of a computer system. Using these new concepts, Control Data could see a machine that would execute one million instructions per second, a faster machine than any presently available computer.

During this initial development period, the transistor industry began to herald the coming of a new and faster transistor. This item, the silicon transistor, was much faster than the old germanium and made greater computer speeds easily available. However, for Control Data it meant a great deal more. Since Control Data had developed new concepts to increase computer system speeds, the new transistor was like frosting on the cake. By using both the new transistor and the newly developed concepts, Control Data now had a machine that would execute three million instructions per second, unapproachable by any other computer, even the "paper tigers". With a computer that could operate at this speed, Control Data had the machine, the 6600, that could handle with ease the particular type of computer problems that were present.

The development of the 6600 Computer was successfully culminated with the delivery of the first 6600 Computer System to the Lawrence Radiation Laboratory at Livermore, California, in June of 1964. The development of the second 6000 Series Computer, the 6400, was started during 1964. The 6400 uses the same basic conceptual changes that were developed for the 6600, the basic difference being a unified central processor rather than one using the principle of concurrent operation.

The conceptual changes used in the 6600 and 6400 are easily understood if one looks at the operation of the typical computer. These computers are designed with extremely powerful and capable arithmetic computing hardware. A great deal of effort is made to make the hardware multiply, divide, and handle computing situations at very high speeds. Ordinarily, the typical computer is called upon to handle the computation of a program and also the processing of its input/output requirements. Therefore, the conclusion that can be made is that, although a computer can be very powerful and capable arithmetically, the sophisticated hardware is not actually doing the job it should be doing most of the time. We can say most of the time because the ratio of input/output time to actual computation time is very unbalanced. This can be illustrated as shown in figure 1.

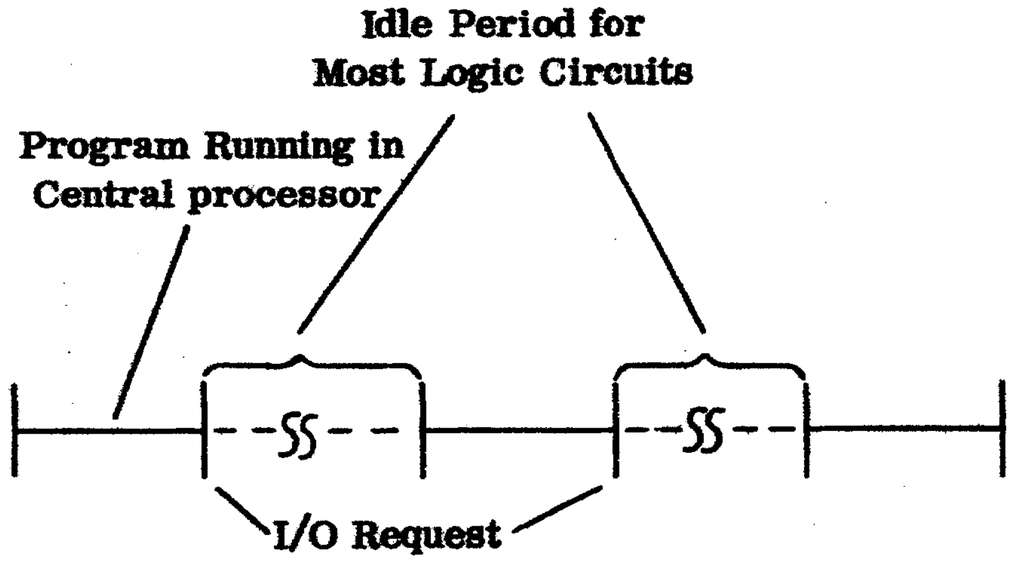


Figure 1. Ratio of I/O Time to Actual Computational Time

In order to point out the awkwardness of this situation, we shall look at an analogy. Assume the president of a large corporation, who is trained and capable of making very high-level decisions, is forced to spend a great deal of his time doing the routine paperwork leading up to and after the big decision. Obviously, he is not using his talents very economically. However, we can be sure that this company president would delegate the routine paperwork to less capable people while he spent his time making the big decision. Of course, it must be understood that this argument holds true only if there are enough decisions that have to be made.

Keeping this analogy in mind, consider that if the central processor could be isolated from the rest of the system, it would lend itself more to pure computing and also provide for faster overall speed in program turn-around. In order to be able to isolate the arithmetic section of the computer, there must be some other area of the machine that is capable of processing input/output independently of operations in the arithmetic section. If successful isolation of the arithmetic section can be accomplished, a tremendous opportunity arises to create an extremely fast central processor. Remember, this central processor (arithmetic section) never need worry about I/O processing.

In the 6000 Series Computer System this is exactly what has been accomplished. The central processor has been isolated from the routine work of I/O processing.

In the 6000 Series Computer there is a very fast central processor unit (CPU) with its own high-speed memory (up to 131K), and ten independent peripheral processor units (PPU's), each with its own 4K memory (figure 2). The function of the CPU in this system is that of high speed computing while the PPU's are

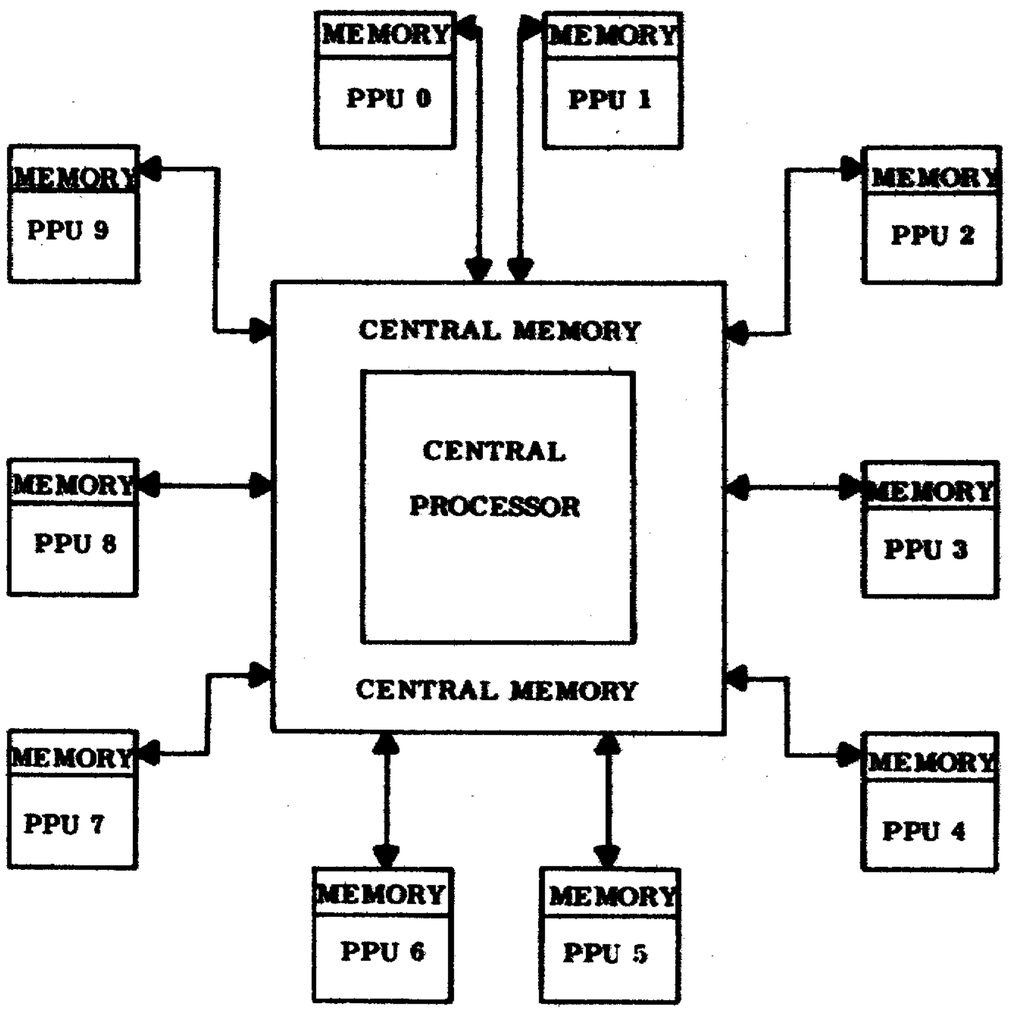


Figure 2. 6000 Computer System
General Block Diagram

assigned the task of processing all of the necessary I/O requests and doing simpler types of problem solving.

Because the 6000 Series Computers have a multi-computer configuration they can handle several programs simultaneously, rather than singly which was the case in the other kind of computers. Because of this fact we must look at the typical operation of these machines in a different light. In doing this we can see how the new 6000 Series Computers conceptual differences permit the handling of programs in a more economical way.

We shall assume a group of programs; the first program is loaded into the computer and execution begins. An I/O request is encountered. At this time the typical 6000 Series Computer could immediately start the second job running in the central processor and assign one of the ten peripheral and control processors the job of processing the input or output request (see figure 3)

Using this scheme, the I/O gap has been removed from the central processor. This means that the central processor is using its capabilities almost 100% of the time.

PERIPHERAL AND CONTROL PROCESSORS

There are ten identical peripheral and control processors (PPU's) associated with the basic 6000 Series Computer System. Each is independent of the other and each has its own 4K, 12-bit memory. Each is capable of communicating with the central memory, the 12 I/O channels, and the real-time clock. Each processor is capable of executing a 64-instruction repertoire. The scheme under which the 10 PPU's operate is one which makes full use of common arithmetic and logical circuits on a timing sharing basis.

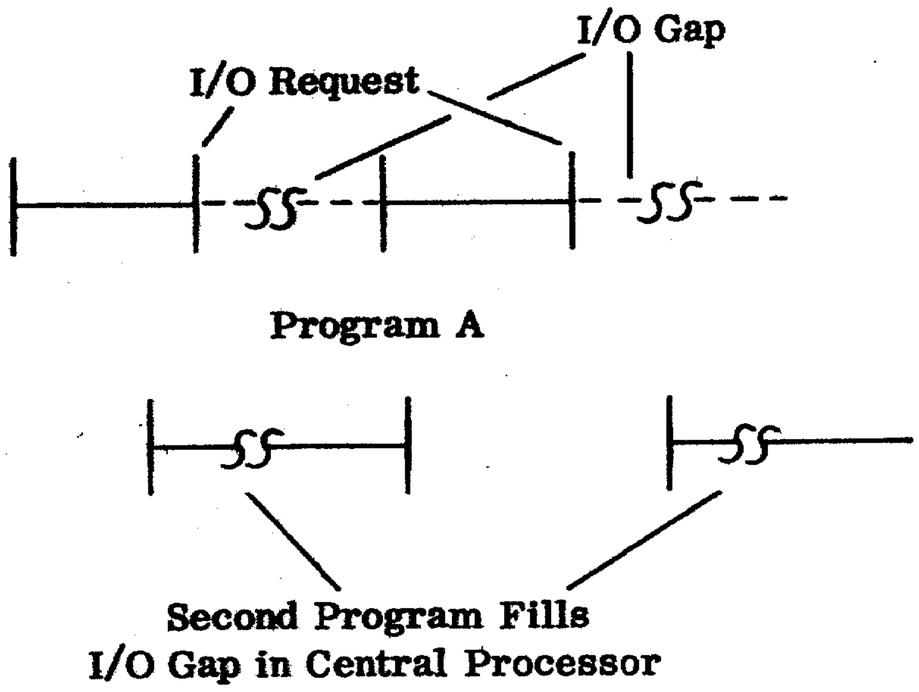


Figure 3. Sequence of Concurrent
Operation of Two Jobs

If we consider the fact that most computers do not make full use of their arithmetic and logical hardware because of memory wait time, the possibility of time-sharing common hardware becomes evident. Consider a single computer. An instruction is acquired from the memory specifying an arithmetic operation to be performed. However, before this operation can be performed, the operand needed must be read from memory. During this access time, the arithmetic hardware is idle. If we were to allow a second computer to use the arithmetic hardware during this period of time, we would be increasing the economic use of the hardware. If the arithmetic and logical hardware were fast enough, couldn't three other computers use it during the memory wait time of the first computer? Why not nine others?

Keeping the above statements in mind, consider computer operation in general. In any computer, the contents of various registers are necessary to control the orderly execution of a program. For instance, a program address register (P) is necessary to keep the program in sequence. An accumulator (A) is necessary to maintain the latest results of some program step.

Usually some sort of register maintains the translation of the latest operation code for extended control of the execution of the present program step. Thus,

it can be seen that we can logically link 10 identical sets of P registers, A registers, and translation registers with one common set of arithmetic and logical hardware. The 6000 peripheral processors do just this.

In the 6000 PPU's each processor has an 18-bit accumulator (A), a 12-bit program register (P), a 12-bit auxiliary accumulator (Q), and a 9-bit translation register (K). All of these registers share a common set of arithmetic hardware. Each set of registers represents one program and is allowed to use the arithmetic hardware for 100ns during each μsec. A circular shifting process is used to move each processor into position to use the arithmetic and logical section. This shifting network is called the barrel. The area where the arithmetic and logical hardware is located is called the slot. Refer to figure 4 for a detailed picture of this scheme. Slot time for any particular processor occurs once every μsec (one major cycle).

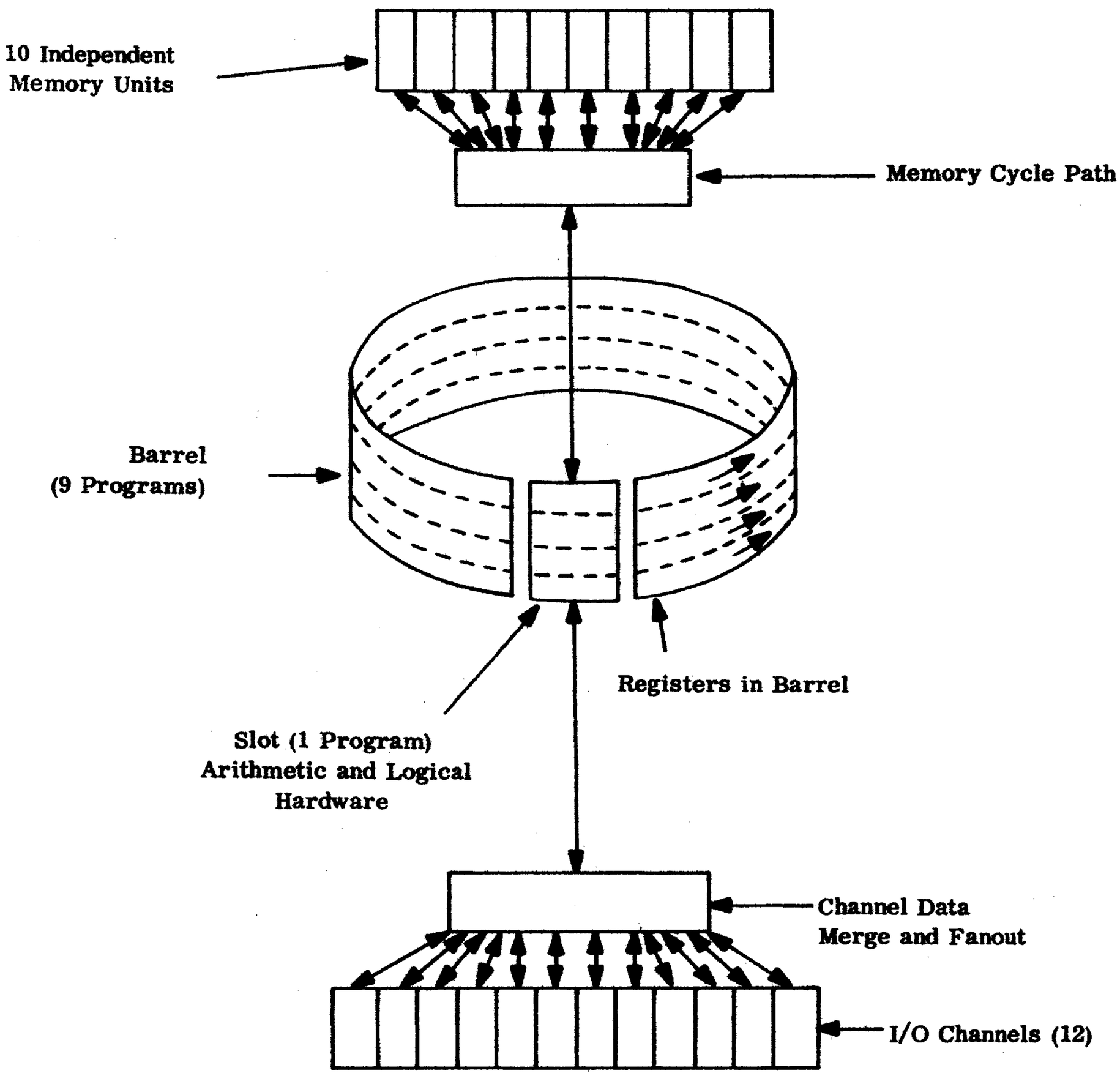


Figure 4. PPU Block Diagram

All arithmetic and logical operations are executed in the slot. Also during the slot time communication with central memory and the I/O channels is accomplished.

PPU DATA CHANNELS

The 10 6000 Series peripheral processors are connected to 13 independent data channels. Twelve of

these data channels are bidirectional and can be connected to I/O equipment. The 13th is permanently connected to the real-time clock and is used specifically for monitoring the clock. The method used to connect the data channels to the peripheral processors allows any processor to use any channel. This scheme allows flexibility in handling I/O equipment.

PPU PHYSICAL CHARACTERISTICS

The 10 processors are identical. One 4096 12-bit magnetic core storage accompanies each processor. The major cycle time is 1000ns and the minor cycle time is 100ns. Twelve I/O channels are common to all processors and all channels are 12-bit bidirectional. The real-time clock uses one μsec per count, 4096 counts per clock period. The instruction set includes logical, branch, I/O, central memory access, central processor access, direct and indirect addressing, and indexed addressing instructions.

Average instruction execution time is two major cycles. Execution times for each instruction are listed in table 1.

Table 1. Peripheral and Control Processor Instruction Times

Octal Code	Name	Time (Major Cycles)	Octal Code	Name	Time (Major Cycles)
00	Pass	1	42	Subtract ((d))	3
01	Long jump to m + (d)	2–3	43	Logical difference ((d))	3
02	Return jump to m + (d)	3–4	44	Store ((d))	3
03	Unconditional jump d	1	45	Replace add ((d))	4
04	Zero jump d	1	46	Replace add one ((d))	4
05	Nonzero jump d	1	47	Replace subtract one ((d))	4
06	Plus jump d	1			
07	Minus jump d	1	50	Load (m + (d))	3–4
			51	Add (m + (d))	3–4
10	Shift d	1	52	Subtract (m + (d))	3–4
11	Logical difference d	1	53	Logical difference (m + (d))	3–4
12	Logical product d	1	54	Store (m + (d))	3–4
13	Selective clear d	1	55	Replace add (m + (d))	4–5
14	Load d	1	56	Replace add one (m + (d))	4–5
15	Load complement d	1	57	Replace subtract one (m + (d))	4–5
16	Add d	1			
17	Subtract d	1	60	Central read from (A) to d	min 6
			61	Central read (d) words from (A) to m	5 plus 5/word
20	Load dm	2			
21	Add dm	2	62	Central write to (A) from d	min 6
22	Logical product dm	2	63	Central write (d) words (A) from m	5 plus 5/word
23	Logical difference dm	2			
24	Pass	1	64	Jump to m if channel d active	2
25	Pass	1	65	Jump to m if channel d inactive	2
26	Exchange jump	min 2	66	Jump to m if channel d full	2
27	Read program address	1	67	Jump to m if channel d empty	2
30	Load (d)	2	70	Input to A from channel d	2
31	Add (d)	2	71	Input (A) words to m from channel d	4 plus 1/word
32	Subtract (d)	2			
33	Logical difference (d)	2	72	Output from A on channel d	2
34	Store (d)	2	73	Output (A) words from m on channel d	4 plus 1/word
35	Replace add (d)	3			
36	Replace add one (d)	3	74	Activate channel d	2
37	Replace subtract one (d)	3	75	Disconnect channel d	2
40	Load ((d))	3	76	Function (A) on channel d	2
41	Add ((d))	3	77	Function m on channel d	2

Earlier in this chapter it was stated that one of the basic concepts of the 6000 Series Computers is the isolation of the central processor. Taking advantage of this isolation principle to create a fast CPU could take many forms. If overlapping of arithmetic operations could be developed where no linkage problems are present, this would mean that concurrent computing of several steps of the program would be possible. Of course, in order to be able to have several arithmetic operations occurring at once, a great supply of operands must be available. Operands could be provided by having a multiple set of accumulators that are used directly by the arithmetic section and have a path to the memory which is independent of the instruction set. Considering that our hypothetical CPU is operating in the fashion described, it certainly would be able to execute a great number of instructions at high speeds. Therefore, we must provide the CPU with a good supply of instructions free of memory. This can be done by having a stack of transistor registers, much like an internal memory, that could be used to draw instructions from.

At this point a classification of the difference between a 6400 and a 6600 computer must be made. The two computers have the same configuration and concept of operation. However, in the 6600 central processor instructions can be executed concurrently where no linkage problems exist, while in the 6400 central processor the execution of instructions is sequential, that is, one after the other.

Therefore, the preceding description of hypothetical possibilities for increasing the speed of the central processor holds true for both computers, with the exception of the concurrent execution of instructions unique to the 6600. Also, the 6400 does not have an instruction stack since, with sequential execution of instructions, there is always sufficient time to acquire the next instruction.

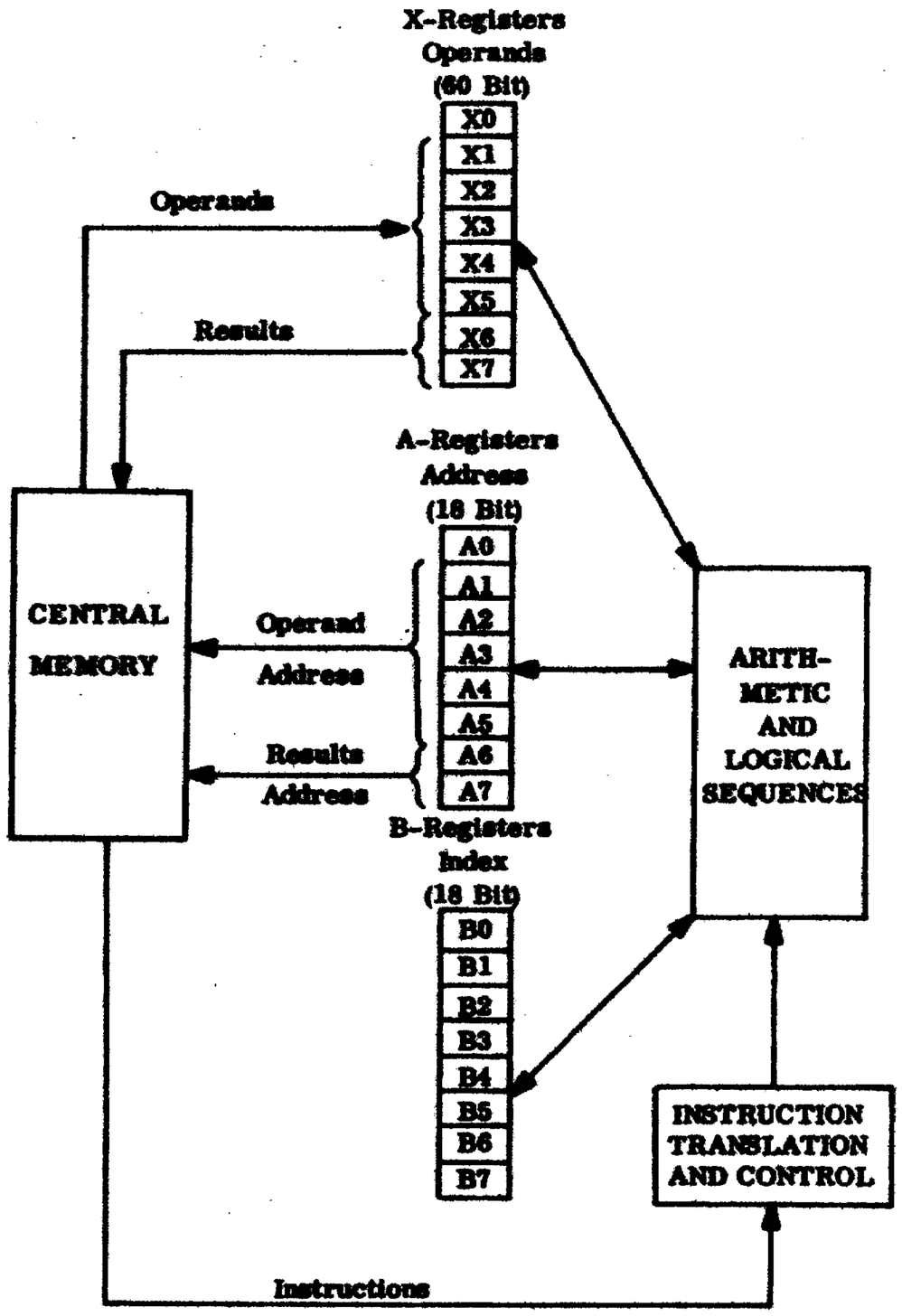


Figure 5. Central Processor Block Diagram

CENTRAL PROCESSOR OPERATION

OPERATING REGISTERS

In order to facilitate the execution of each program step and to reduce memory access times, operands used during the execution of an instruction will come from 24 flip-flop operating registers that will be loaded and stored under program control.

The central processor's 24 operating registers are divided as follows:

- Eight 60-bit X registers that hold operands used for computation. The X registers have a direct access path to central memory.
- Eight 18-bit A registers. These registers will be used for loading and storing operands in the X registers. These registers are known as the Address registers.
- Eight 18-bit B registers.

Changing an A register causes a memory reference for an associative X register. That is to say, changing A1-A5 causes a memory read and X1-X5 receive the operand. Changing A6 or A7 causes a memory write and the operand in X6 and X7 is stored. A0 and X0 are not considered in this memory scheme. The eight 18-bit B registers are used as index registers. All of the above-mentioned registers are unique to the central processor and are not associated with the peripheral processors in any way.

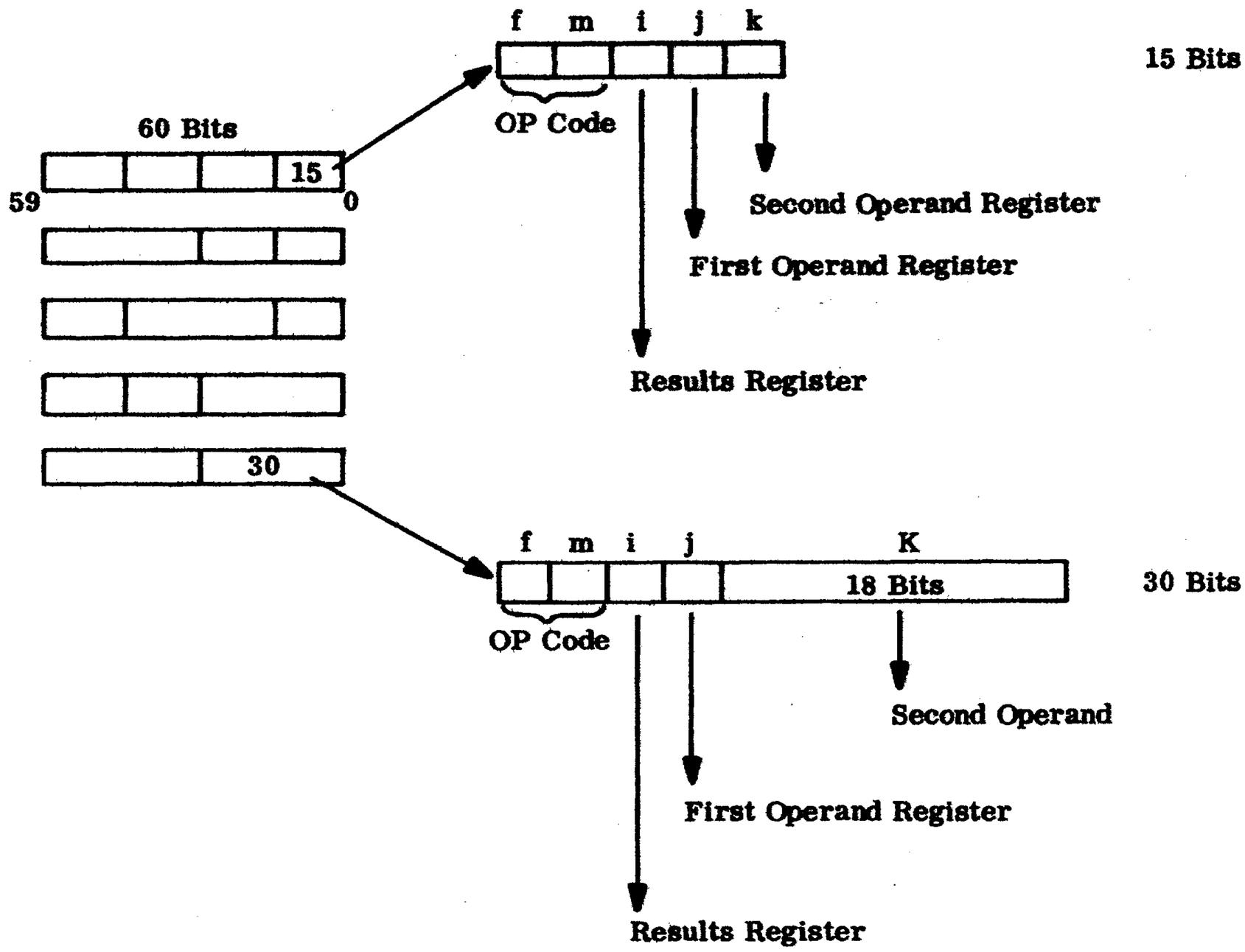


Figure 6. Instruction Formats

INSTRUCTION FORMAT

The basic format of a 6000 Series Computer instruction is arranged in such a manner as to facilitate easy understanding of its operation. Figure 6 illustrates the format of the instructions and also shows all combinations of 15- and 30-bit instructions possible in a 60-bit word. In examining the instruction format, we find that the upper six bits are the operation code. The next three bits, the i portion, designate the operating register where the results of execution of a program step go. The j and k portions designate those operating registers where source operands are found which are needed to execute the program step. If an instruction is a 30-bit type, we find that an 18-bit constant K is part of the instruction and serves as one operand. In this case, j and K serve as the two operands necessary to execute the program step. Always remembering that i is the result register and j and k (or K) are source operands, we find that interpreting the code is quite easy. For instance, the multiply instruction reads:

40 Floating product of Xj and Xk to Xi

The 40 is the operation code and the results go to an X register, one of eight. Both source operands come from X registers. The coded form of the instruction is shown below.

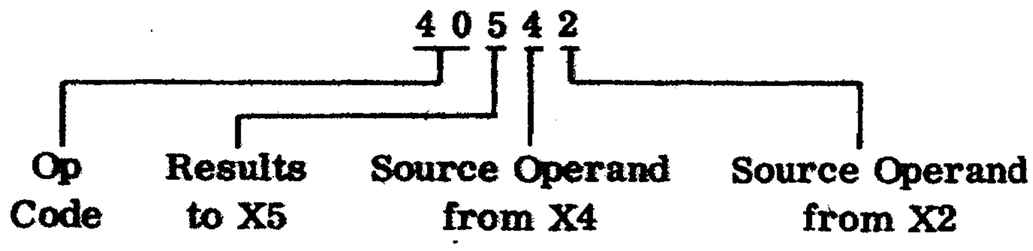


Figure 7 illustrates a short program using symbolic form of coding. This program also gives an example of the manipulation of operands with the operating registers.

Operation	Result
A1 = B1 + K	Load X1 from memory location B1 + K
A2 = B2 + B0	Load X2 from memory location B2 + B0
X6 = X1 · X2	Multiply X1 by X2 and store results in X6
A6 = B5 + K	Store X6 in memory location B5 + K

Figure 7. Symbolic Coding

EXCHANGE JUMP

A peripheral and control processor starts the first central processor program and any new program running in the central processor by executing an Exchange Jump instruction. This scheme is necessary since the central processor is like an island. That is, there are no logical external connections to any part of the central processor controls that can affect it. Only through programming can the central processor be reached. The exchange jump method of starting programs in the central processor is ideal since it allows the interruption of a program running in the central processor and the recovery of the same program at a later time.

One situation where we would want to do this is when the program presently residing in the central processor has reached an idle period because it requires some input or output processing. At this time, an exchange jump would allow us to start a new program running in the central processor and would also record the necessary information needed to restart the original program when the input or output processing had been completed.

In order to get a basic understanding of how the exchange jump package works, remember that any program running in the central processor is under the control of the operating and control registers. These registers, such as the Program Address register (P), keep track of the sequence of an instruction; the X, B, and A operating registers keep track of operands being used for the execution of program steps; and the Reference Address register (RA) keeps track of the relocation point of the present program. There are other important registers, of course, but these serve as an example of what is meant.

A program running in the central processor can be stopped and restarted without difficulty if a record can be made of the control and operating registers at the time of the interrupt. As for the new program that we want to start executing, we must supply new values for the control and operating registers that the new program will be using.

Keeping the above statements in mind, we shall have a peripheral processor store into the central memory new values for the control and operating registers that will be used by the interrupting program. We shall call this area of central memory the exchange jump package (figure 8). Of course, the instructions for our new program must also be located in the central memory before we execute the exchange jump (figure 9).

When we have stored our exchange jump package in central memory, the PPU will execute the exchange jump. This will cause the exchange of the contents of the operating and control registers, now being used by the present program, with new values. These new values will come from the exchange jump package and be placed in the operating and control registers. The old program values used to represent the conditions and control registers are safely stored in the exchange jump package. Another exchange jump will restore these values in the operating and control registers whenever we want to continue execution of the original program.

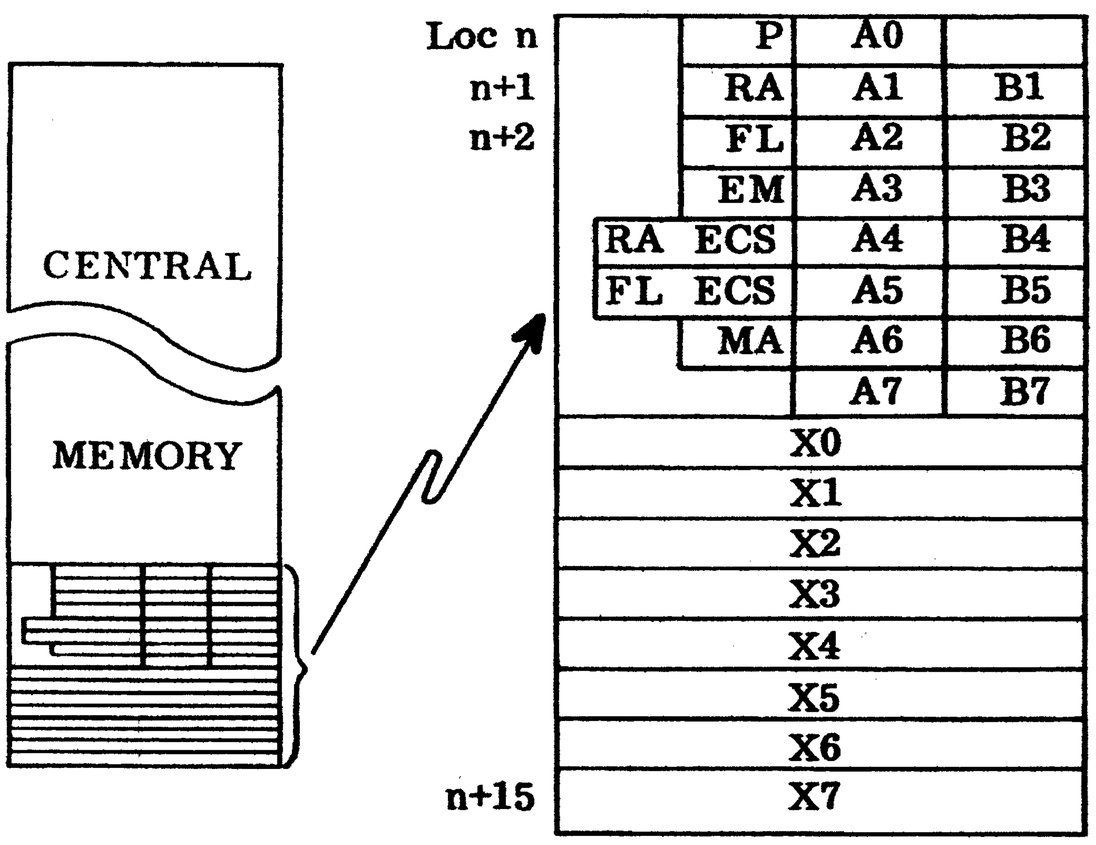


Figure 8. Exchange Jump Package

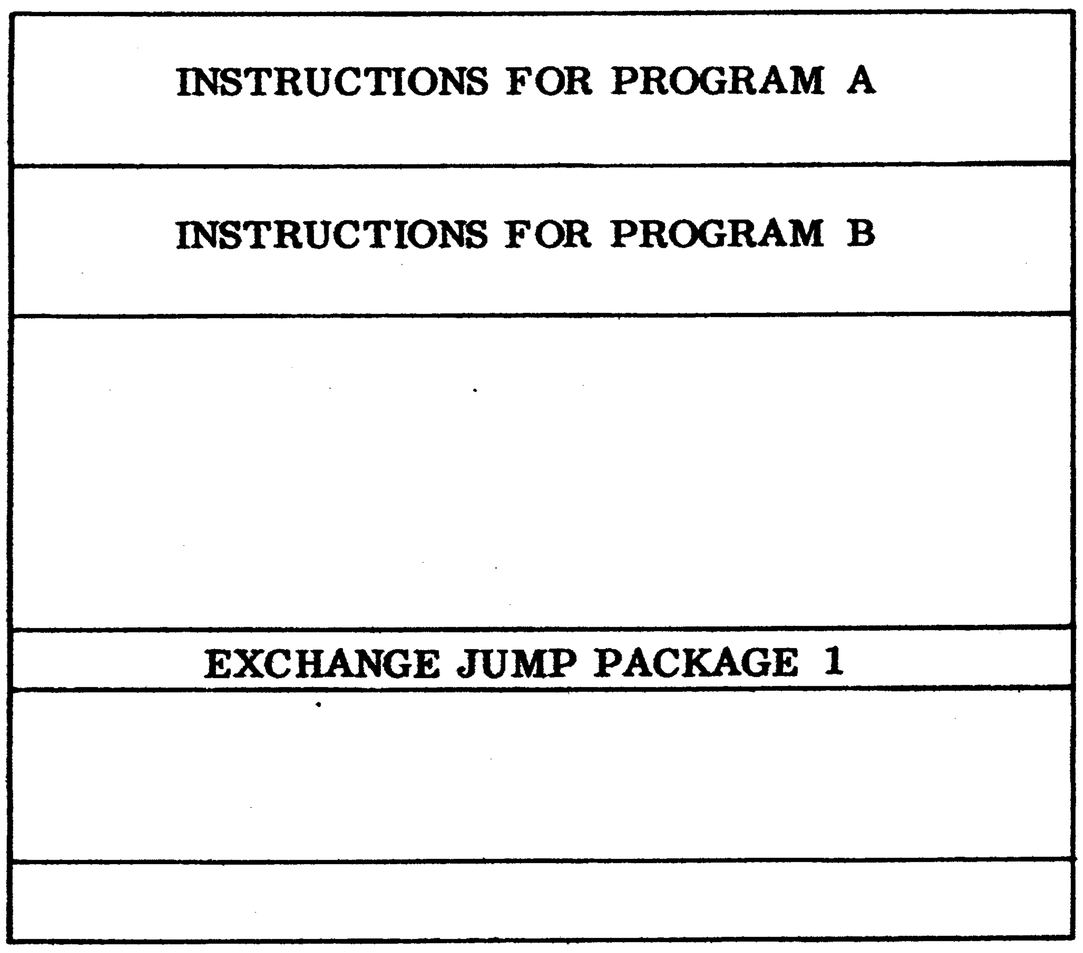


Figure 9. Program Distribution in Central Memory

PHYSICAL CONCEPTS

Transistor-coupled logical circuits are used throughout the 6000 Series Computer. The circuits are cooled by means of refrigeration units contained within the mainframe. Memory for the 6000 Series Computers is magnetic cores, arranged in modules that contain 4096 12-bit locations. Data and control between chassis and to external equipment are pulsed using transformers feeding coaxial cables. Internal data and control flows over twisted pair lines, and both ends of one of the wires are terminated to ground.

LOGIC CIRCUITS

A technique called cordwood packaging is used when building the individual modules that make up the 6000 Series logic circuits. The logic module will average 64 transistors and has a 30-pin male connector for signal distribution (figure 10).

The transistors used are NPN type, called silicon epitaxial pacivated transistors. These transistors switch in three to five nanoseconds and operate in saturated circuitry. The logical circuit that is contained on a module may be unique or may be a standard circuit that can be repeated many times within the overall computer. The individual modules are pluggable into a 30-pin female connector that is mounted on the chassis. The individual connectors are connected together using point-to-point wire techniques. This forms a thick wiring mat on the back of each chassis (figure 11). Specific modules can be located on the chassis by coordinates. The vertical edge of the chassis is labeled A-R and the horizontal bar is labeled 1-42. There are test points on the front plate of each module for ease of monitoring the internal signals of the modules.

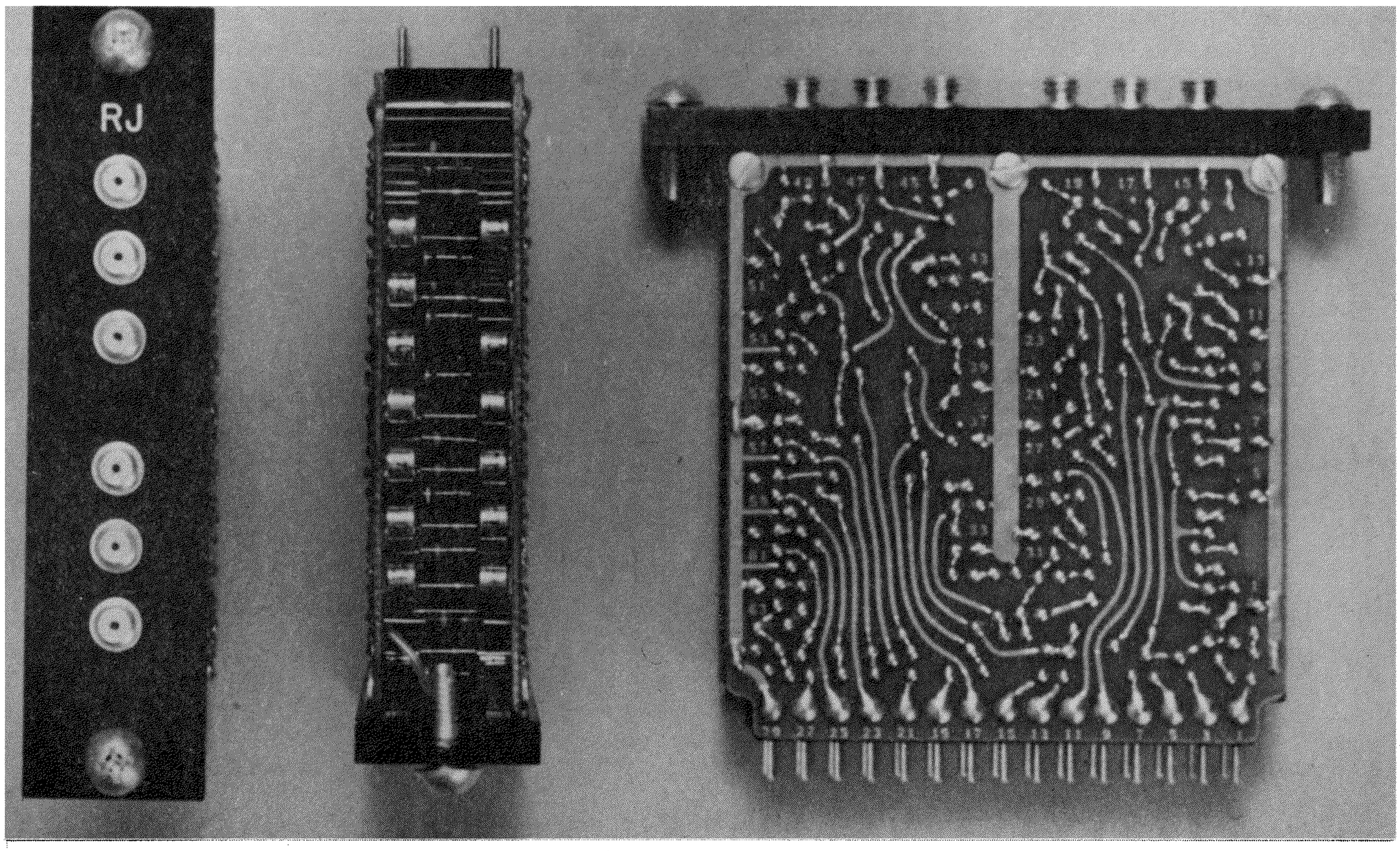


Figure 10. 6000 Logic Module

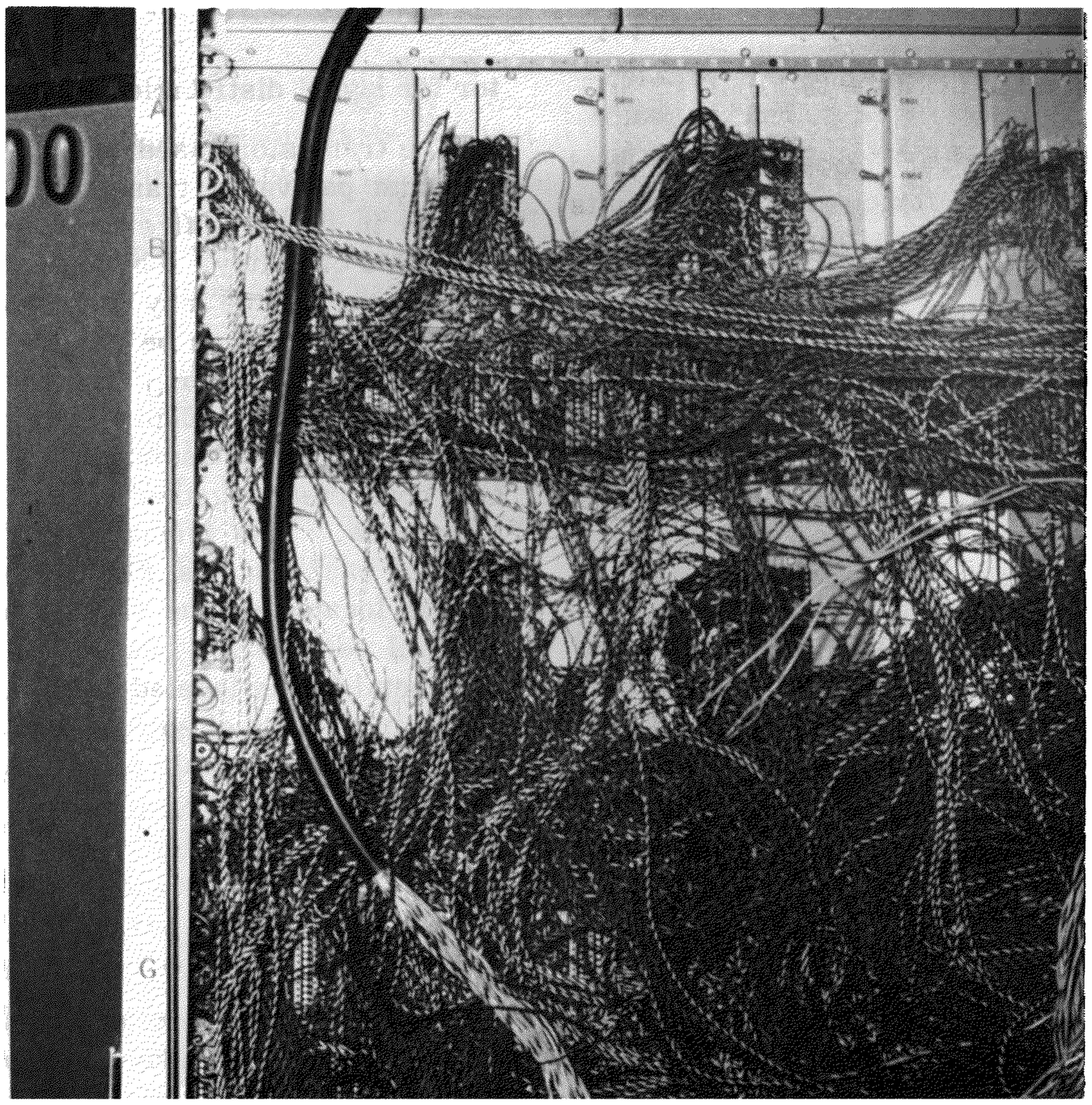


Figure 11. Back of Chassis Wiring

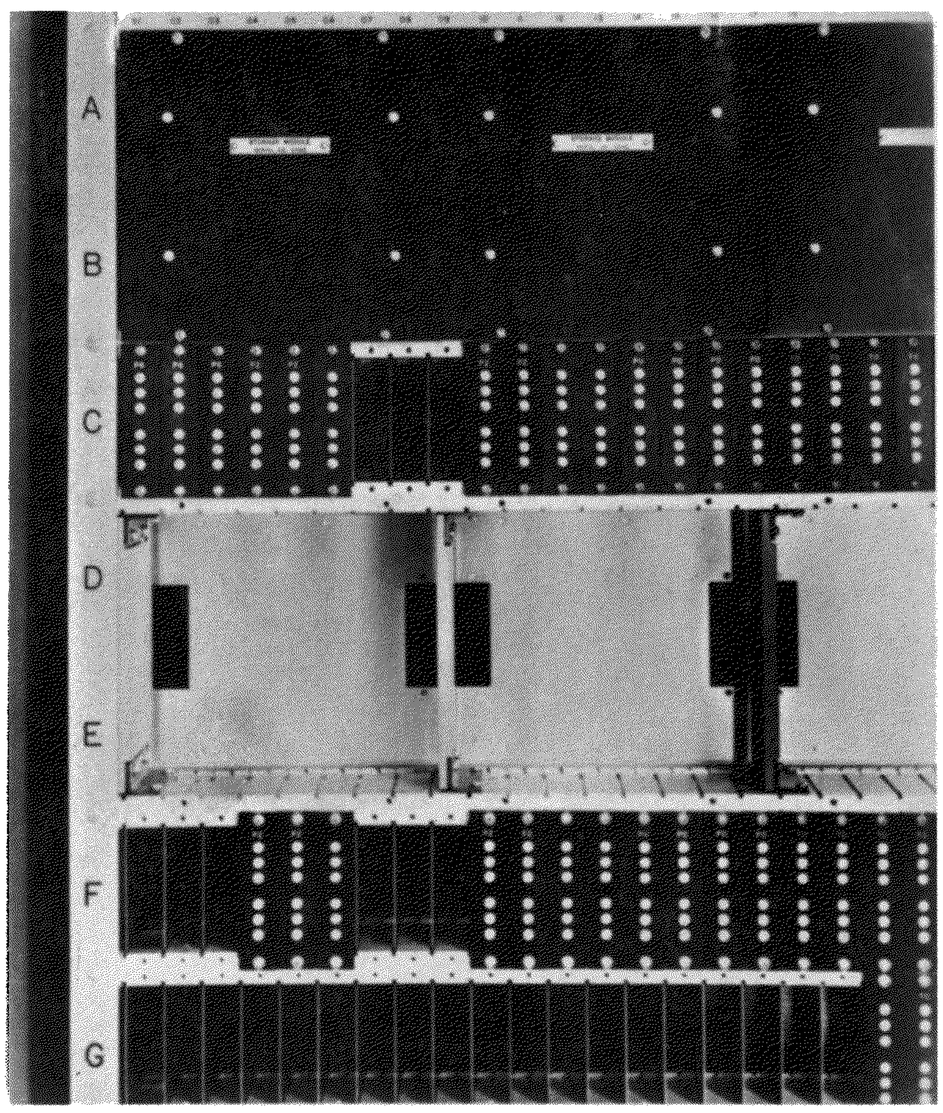


Figure 12. Typical 6000 Chassis

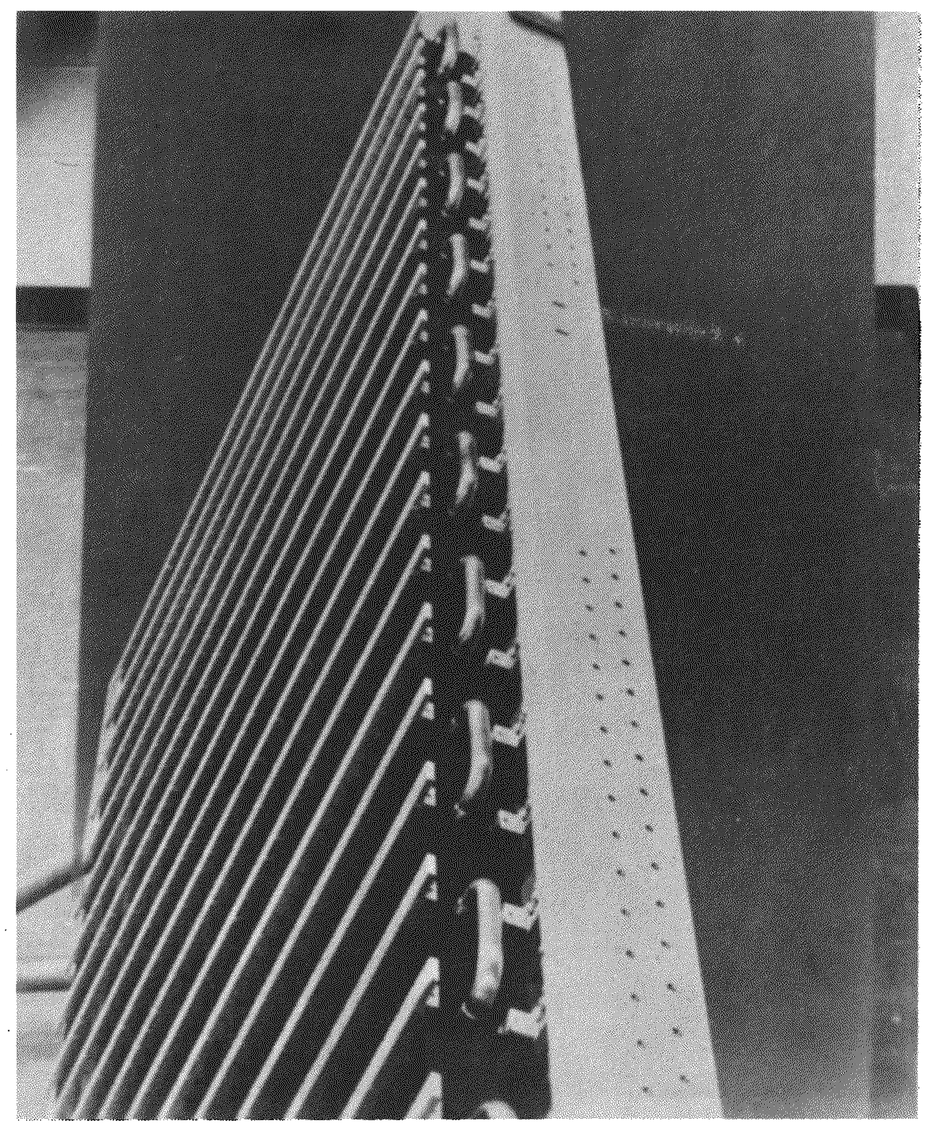


Figure 13. Refrigeration Tubes

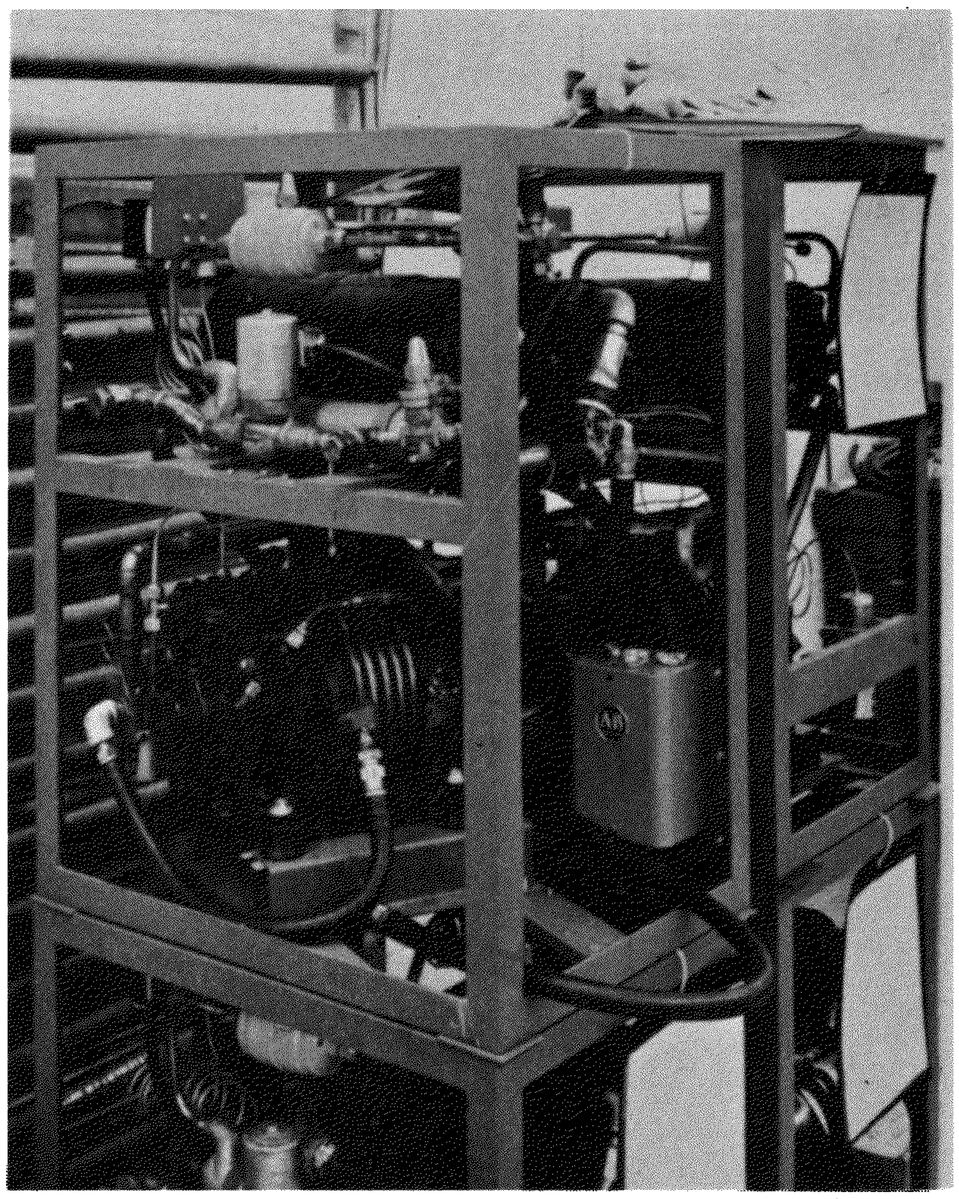


Figure 14. Refrigeration Unit

COOLING

Between each row of modules on a chassis is a bar containing a copper tube that carries refrigerant (figure 13). The refrigerant carries heat away from the bar to a closed cycle refrigeration unit located in each wing of the mainframe (figure 14). The heat generated by the modules is transferred to the cold bar by the principles of convection and conduction.

MAGNETIC CORE STORAGE

A unique package, the memory module (figure 15), contains 49,152 magnetic cores used for storing bits of information. The size of each core is 0.012 inch inside diameter and 0.020 inch outside diameter. The cores are arranged on 12 planes, each plane having a 64 x 64 matrix. Each memory module has 4096 addressable locations of 12 bits each. The memory module contains all the drive and inhibit circuits necessary to operate that module. Only an External Address register and sense amplifiers are necessary to operate the module. The memory modules are plugged into the chassis in the same manner as the logic modules and are easily replaced when necessary. One module makes up each independent memory for the peripheral and control processors.

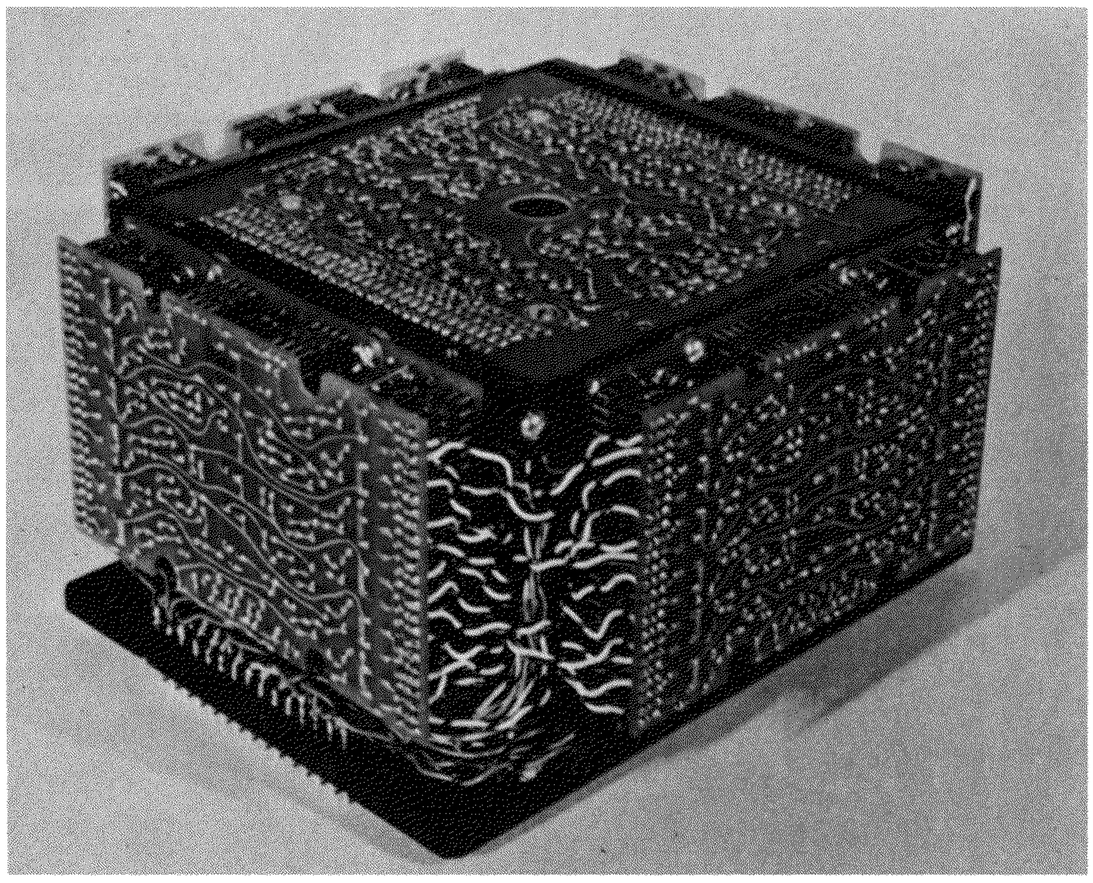


Figure 15. 6000 Memory Module

SIGNAL TRANSMISSION

Because of the extremely high speed of the 6000 Series logic circuits, a method of moving data is used that does not reduce the speed of the operating circuits. This method is transformer-coupled coaxial cables arranged in groups called data trunks, which are located between points where data flows, such as the data trunk between central memory and the operating registers. The trunk consists of two 60-wire coaxial cables, transmitting circuits, and receiving circuits (FFs).

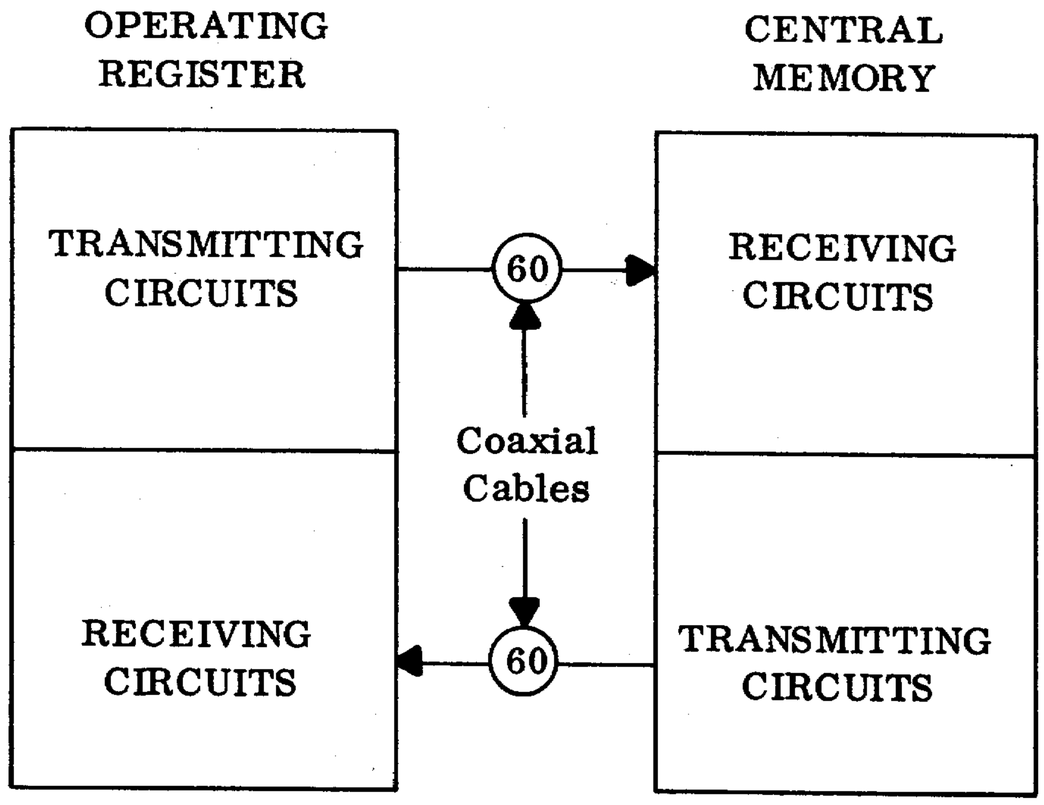


Figure 16. Data Trunk between Central Memory and Operating Registers

Below is a listing of the various components possible with the 6600 and 6400 Computer Systems.

6400 SYSTEM

- 6413 Central Computer with 131,072 words of magnetic core storage, 10 peripheral and control processors with storage
- 6414 Central Computer with 65,536 words of magnetic core storage, 10 peripheral and control processors with storage
- 6415 Central Computer with 32,768 words of magnetic core storage, 10 peripheral and control processors with storage

6600 SYSTEM

- 6613 Central Computer with 131,072 words of magnetic core storage, 10 peripheral and control processors with storage
- 6614 Central Computer with 65,536 words of magnetic core storage, 10 peripheral and control processors with storage
- 6615 Central Computer with 32,768 words of magnetic core storage, 10 peripheral and control processors with storage

6400 OR 6600 SYSTEM

- 6416 Augmented I/O Buffer and Control includes 16,384 words (60 bits) of storage and 10 peripheral and control processors with storage. Communicates with central computers via standard channel. While this device is not currently being manufactured, several are in the field.
- 663X Extended Core Storage contains 125K to 1000K words of magnetic core storage (60 bits), depending on the specific model. Also included are couplers and controls enabling communications with 6400 and 6600 central computers and 6416 I/O buffer and control. First word access is three μsec for all models. Transfer rate depends on model (table 2).

Table 2. Extended Core Storage Transfer Rates for Each Core Storage Model

Model	Memory Size	Transfer Rate
6633	125,952	Up to 2,500 Words/Sec
6634	251,904	Up to 5,000 Words/Sec
6635	503,808	Up to 10,000 Words/Sec
6636	1,007,616	Up to 10,000 Words/Sec

In addition to the various options possible with the system, there is also a variety of peripheral equipment available.

6612 DISPLAY CONSOLE

The 6612 unit (figure 17) comprises the display section of the 6000 System. The display console consists of two 10-inch cathode ray tube display units and a manual keyboard containing 50 alphanumeric and special characters.

The console can be selected to display in either character or dot mode. In the character mode, two alphanumeric characters can be displayed for each 12-bit word sent from a peripheral processor.

The three character sizes are small (64 char./line), medium (32 char./line, and large (16 char./line).

In the dot mode, a pattern of dots (graphs, figures, etc.)can be displayed. Dots are generated for 12-bit words containing vertical or horizontal coordinates. Upon receipt of a y coordinate, a dot is painted at that y coordinate and the last previous x coordinate.

Typical operation of a display console in the 6000 Series System allocates one screen for presentation of operator directives. The remaining screen gives the operator status information on the current problem or information on other problems being run. Although none of the registers or memory locations are displayed automatically, a peripheral processor control program can extract register information and central memory contents and send it to a screen for viewing. In addition, a control program can change central memory or register contents and can interrupt, step, or terminate a central processor program from manual input on the console keyboard.

The 6612 Display Console operates under the control of a peripheral and control processor. This processor can operate several displays simultaneously. Multiple units minimize idle time in the system and allow simultaneous debugging and monitoring of many unrelated problems.

6603 DISK FILE

The 6603 Disk File (figure 18) is a mass storage unit for the 6000 Series Computer System. It has the following features:

- Capacity of 448,266,240 bits, or 74,711,040 characters
- Transfer rate of 1.09385×10^6 char./sec (12 bits every 1.83μsec) for the inner zones when controlled by two peripheral and control processors

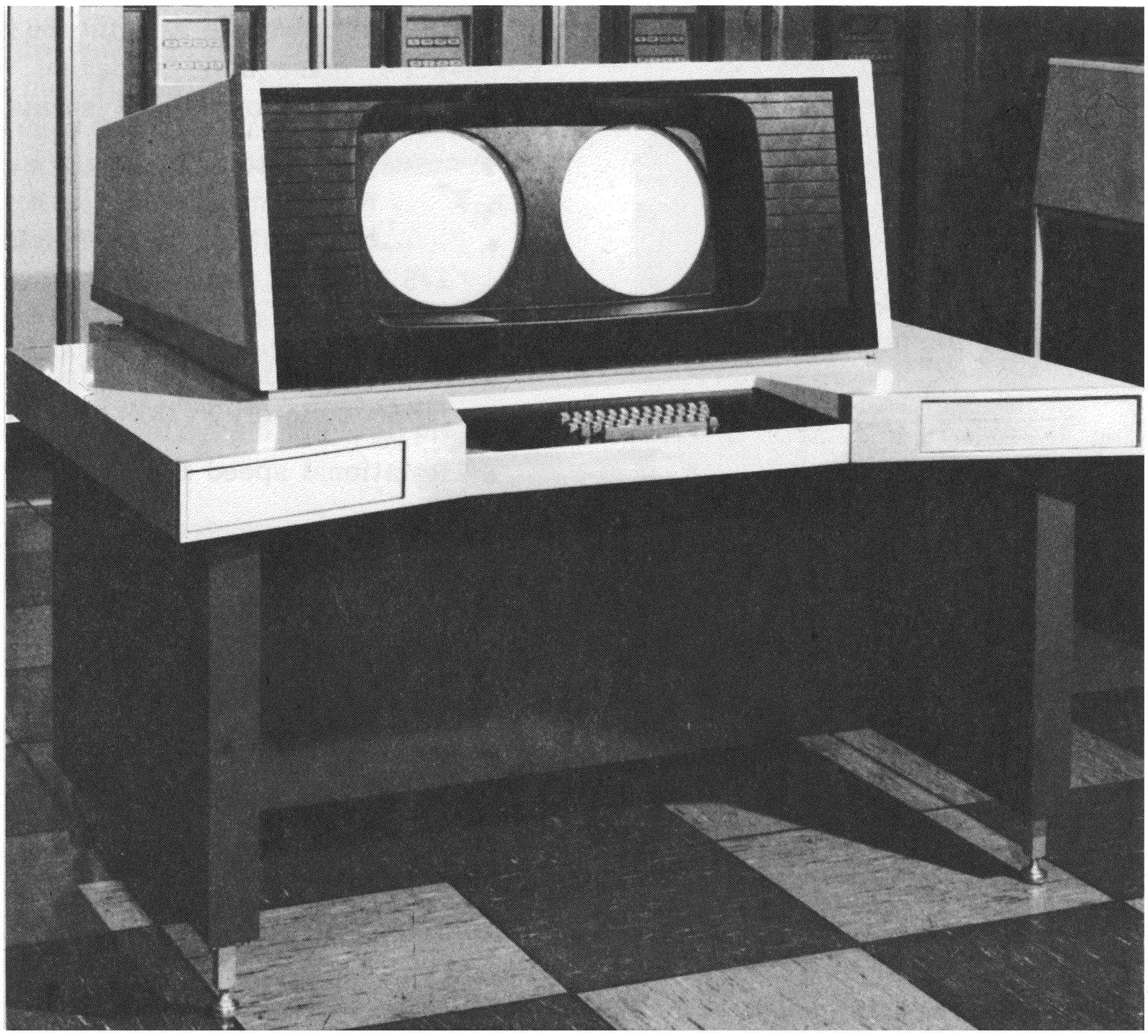


Figure 17. 6612 Display Console

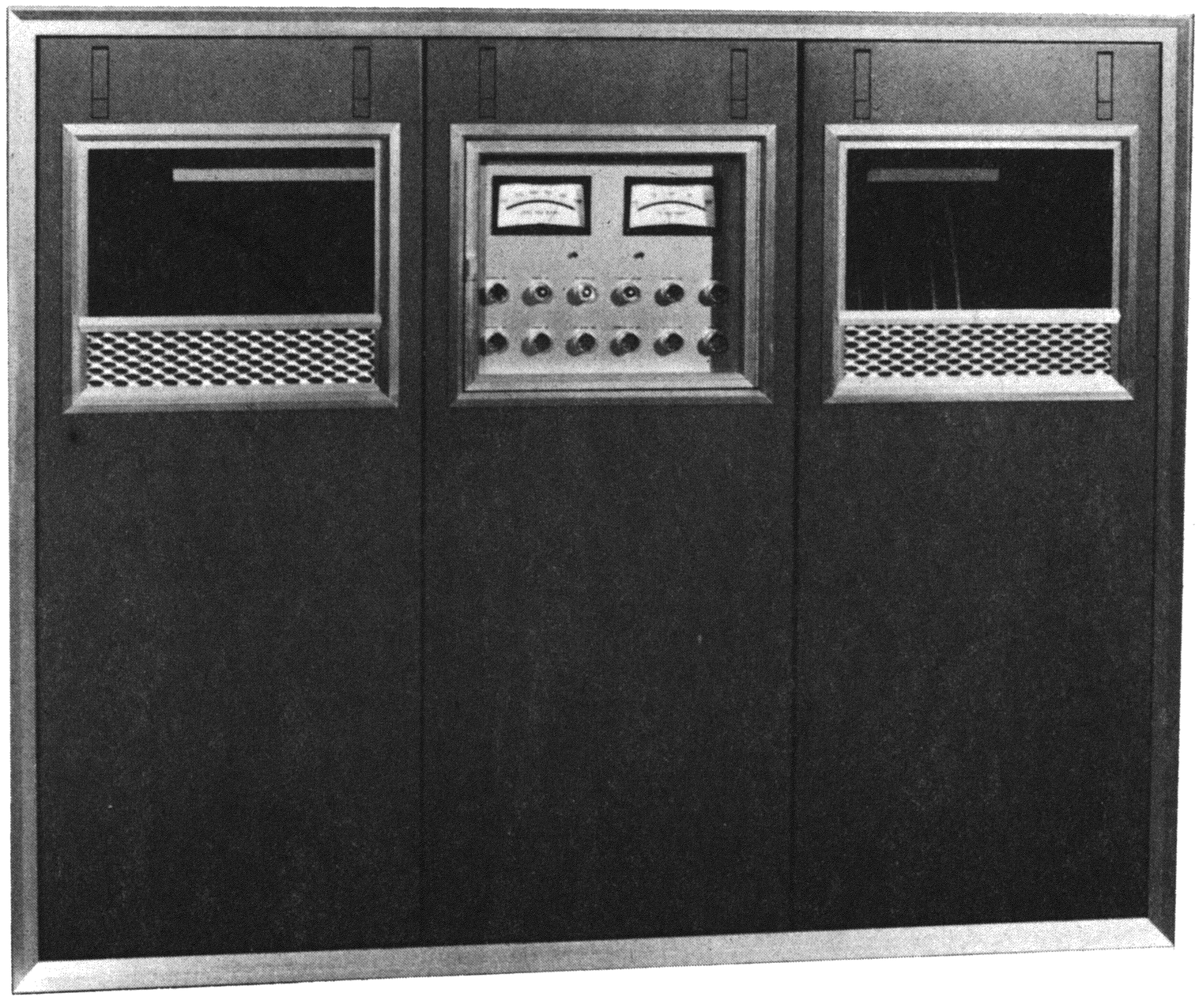


Figure 18. 6603 Disk File

- 1.4×10^6 char./sec maximum sustained rate for outer zones
- 1.2×10^6 char./sec average sustained rate
- Rotational speed of 66.6ms/revolution
- 266.4ms average total random access time
- 96 magnetic heads, four heads per disk surface; all heads are moved in unison by a single positioning arm
- Error checking by program control and read parity checks

6638 DISK FILE SYSTEM

The 6638 Disk File System (figure 19), consisting of the 807 or 808 file and the 6639 controller, is a large-scale mass storage device for use primarily with 6000 Series Computer Systems.

Features of this system are:

- Capacity of 1,054,113,792 (maximum) bits or 175,685,632 (maximum) characters, random-accessed by track (808)
- Transfer rate of 12 bits (parallel) each 1.18μsec (approximately 847kHz) for all data on disk
- Rotational speed 1200 rpm (approximate)
- 163.25ms average random access time (worst case) including positioning, verification, and latency

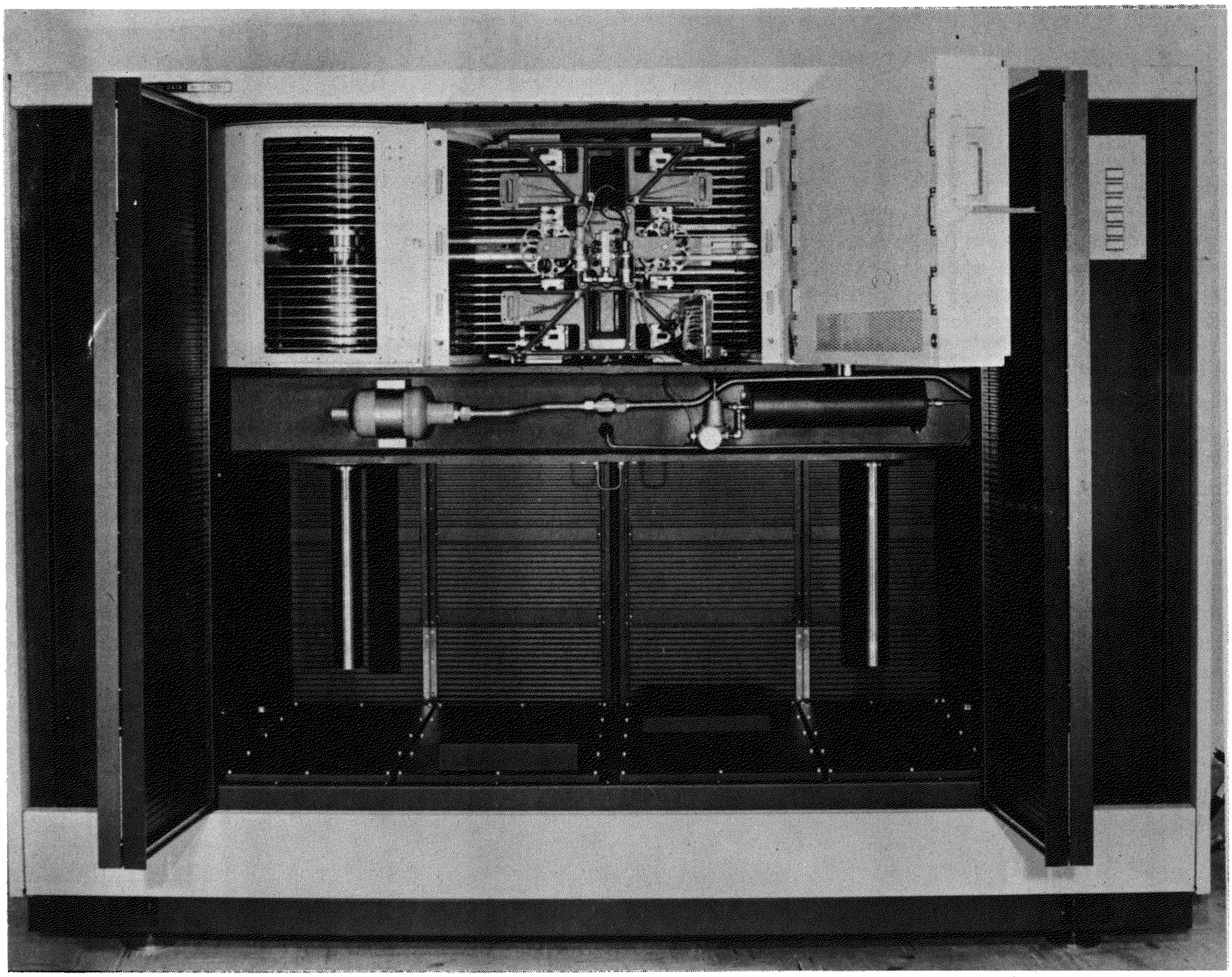


Figure 19. 807 Disk File

- 64 read/write head groups, each containing 12 read/write heads; half of the groups (32) are attached to each positioner, and the positioners are addressed separately so one positioner can be moving to a new position while reading or writing is carried out with the other (808)

- An 807 is physically half the capacity of the 808 and has one positioner instead of two

- Error checking by program control plus read parity check

6622 MAGNETIC TAPE CONTROLLER

This controller permits reading and writing of any one of four model 626 Magnetic Tape Transports. It operates from one standard channel.

626-B MAGNETIC TAPE TRANSPORT

The Control Data 626-B Magnetic Tape Transport (figure 20) is very much like the Control Data 607-B Magnetic Tape Transport. The 626-B is designed specifically as an intermediate storage device for the 6000 Series Computer System. The 626-B tape unit uses one inch tape with 14 tracks per frame (across the tape). The frame format is as follows: tracks 0-5 contain data bits 0-5, track 6 contains parity bit for preceding tracks; tracks 7-12 contain data bits 6-11, track 13 contains parity bit for preceding six tracks.

Information is recorded in the binary mode only. Each frame represents one 12-bit peripheral processor word, split into two characters, with an odd parity bit for each half.

The packing density is fixed at 800 bpi. Tape speed forward is 150 ips. The nominal transfer rate is 240,000 (six bit) char./sec.

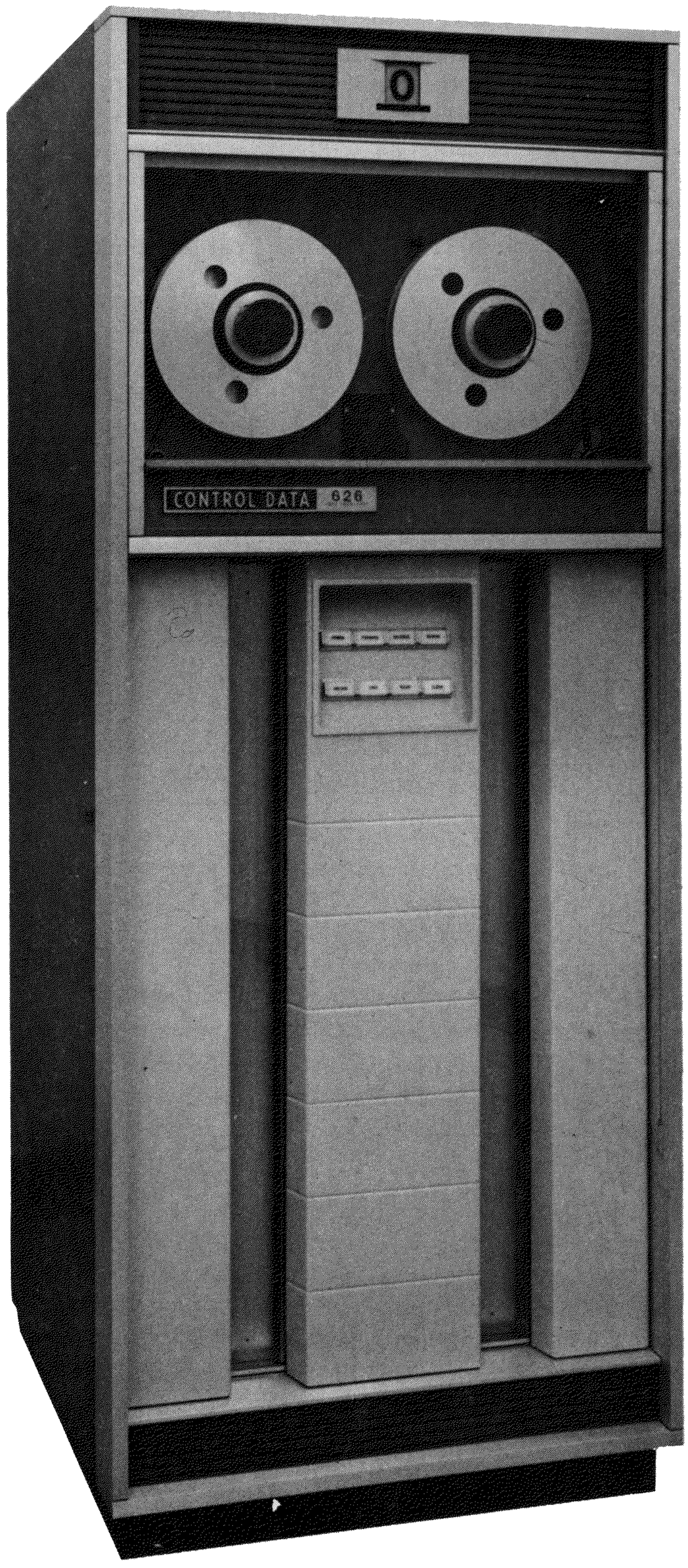


Figure 20. 626-B Magnetic Tape Transport

6681 DATA CHANNEL CONVERTER

The 6681 allows the 6000 Series peripheral processors to communicate with 3000 Series I/O equipment. The converter acts as a signal interface and allows users who presently have 3000 Series computers or equipment to use this equipment with the 6000 Series System. The 6681 must be the first equipment on the channel and can control up to eight pieces of equipment.

In order to allow the utmost in flexibility in configuration of the external equipment, the 3000 Series peripheral controllers are available. These controllers are used for the card readers, card punches, and half-inch magnetic tapes.

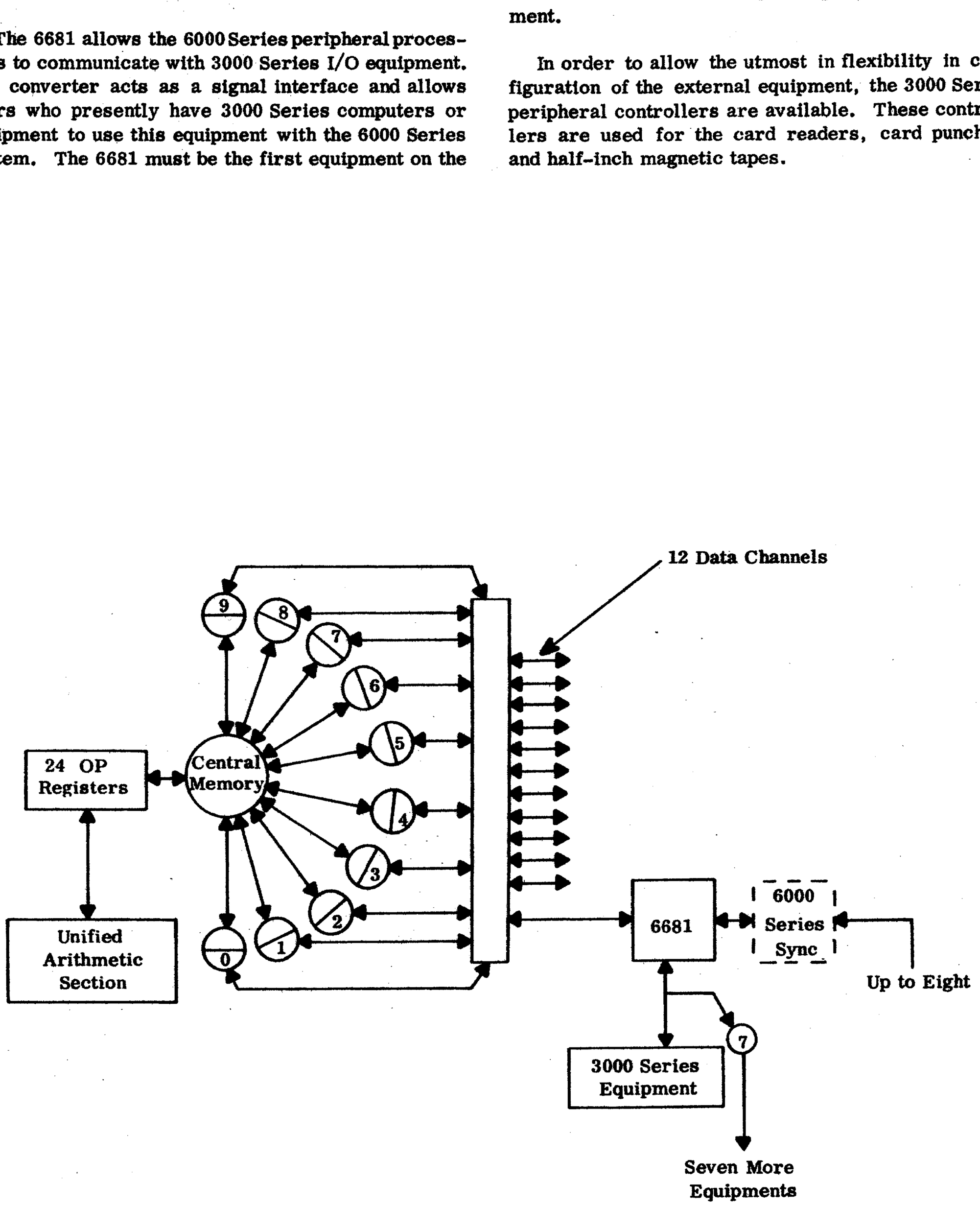


Figure 21. 6681 Data Channel Converter

CHAPTER 2

6000 SERIES LOGIC ANALYSIS

CIRCUITS

The majority of the circuits in the 6000 Series Computers use the inversion property of transistors to obtain logical OR and AND configurations. These, in turn, are connected into flip-flops which temporarily store data. Special circuits drive twisted pair or coaxial cable transmission lines, control storage or readout of data in the magnetic core memory, amplify and detect magnetic tape or disk file head signals, and control other functions.

PACKAGING

Circuits are packaged in modules (figure 10) which use printed circuit wiring. Circuit components are mounted on and between two printed circuit boards in a method sometimes referred to as cordwood packaging. A 30-pin connector fastened at one end of the module allows electrical access to the circuits. The module connector mates with a chassis connector whose pins are wired to other similar circuits. A plate with captive screws is fastened to the module at the end opposite the connector. This plate connects the module to the chassis mechanically and also conducts component heat to the chassis cooling system. The plate carries an identifying number for the module and six test-point terminals. The test points are numbered one through six from top to bottom and are connected to various parts of a circuit to allow circuit operation to be viewed on an oscilloscope.

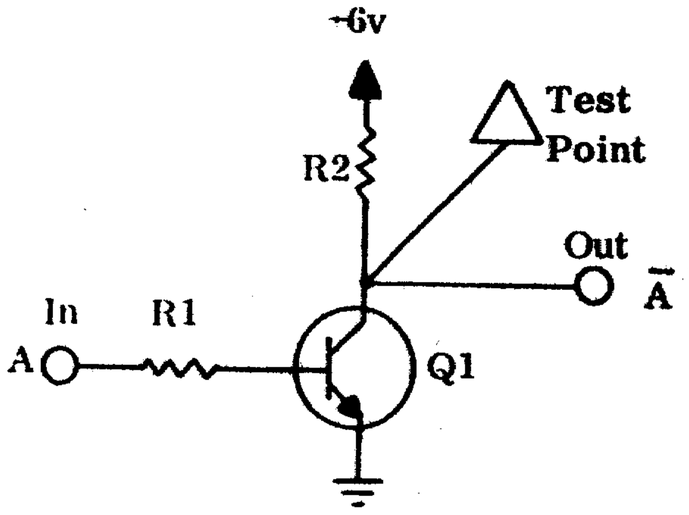
COOLING

Modules are mounted vertically in the chassis in separate compartments. Compartment side walls,

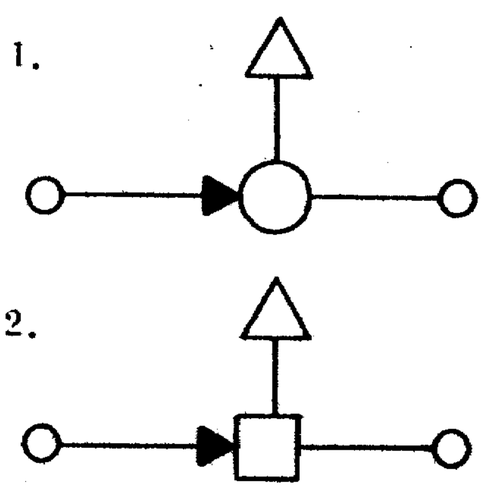
the connector (back), and module plate (front) have a black finish. Rows of modules are separated and supported by metal plates (cold bars) which form the top and bottom of the compartments. A copper tube is imbedded in each bar, and it is connected to a freon refrigeration (cooling) system. Component heat from each module is carried to the cold bars and cooling system by convection and by black-body conduction.

BASIC CIRCUIT ANALYSIS

The basic circuit is a single NPN silicon transistor connected in a grounded emitter amplifier circuit. The transistor operates from a +6vdc supply. The inherent 180 degree phase shift of the transistor provides the basic inversion (or NOT) function. The two signal levels and their logical significance are: +0.2v is one and +1.2v is zero (figure 22).



Basic Inverter



Logical Representations

Input		Output	
+0.2v	1	+1.2v	0
+1.2v	0	+0.2v	1

Figure 22. Basic Inverter

A +1.2 input turns on the transistor and drives it to saturation. The Output signal is then the drop across the transistor, +0.2v. When a +0.2v input cuts off the transistor, the output rises toward +6v but is held to +1.2v by the voltage-dividing effect of the collector resistor and the input resistance of the driven load.

The threshold voltage on the base of the transistor is approximately +0.8v (virtually no conduction below +0.7v) which permits the circuit to reject noise signals up to about +0.5v. Circuit parameters for cutoff and saturation are listed in example 1.

Example 1

	V_B	V_C	I_B (min)	I_C	Average Switching Time
Cutoff	0.2	1.2	0	0	5ns
Saturation	0.8	0.2	1ma	10ma	

AND and OR COMBINATIONS

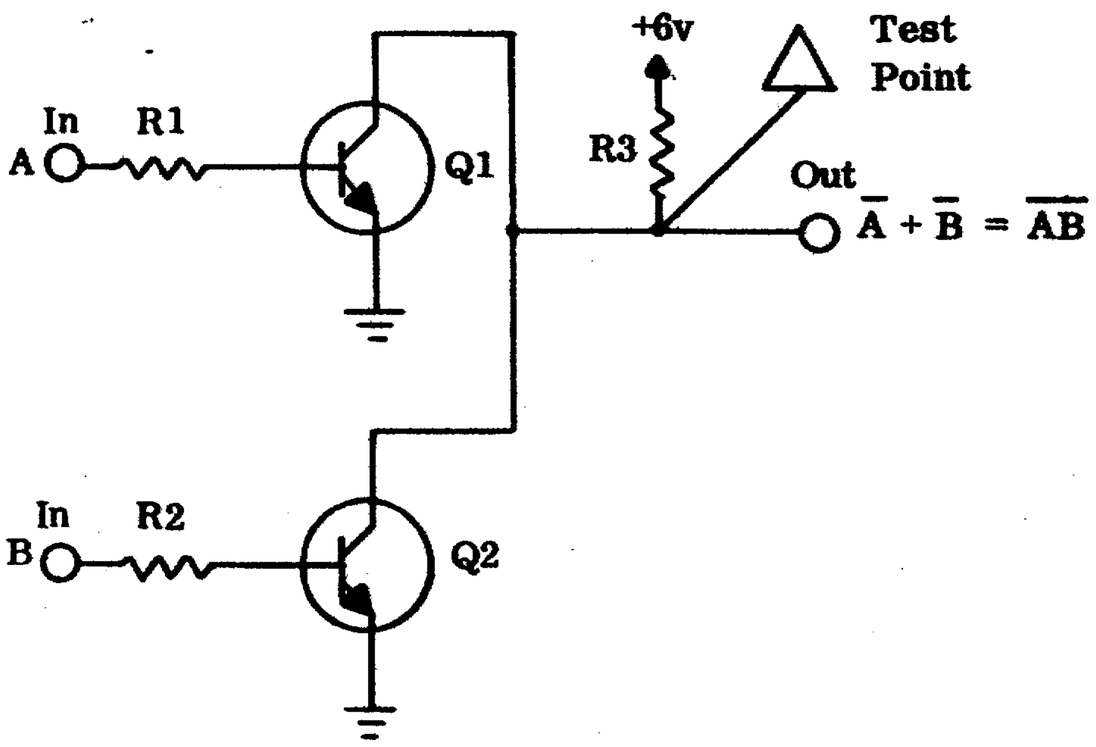
To combine signals from two circuits, the collectors are connected to a common load resistor (figure 23). This provides $\bar{A} + \bar{B} = \overline{(AB)}$ as the Output signal. This method of combining signals is sometimes called NOR logic. The name derives from NOT OR since the circuit consists of an OR with phase inversion or negation of the inputs. Up to six transistors may be connected to a common collector resistor in this way.

The circuit to combine two signals A and B to provide AB as the output appears in figure 24. When either A or B is +1.2v, Q1 or Q2 is biased to saturation and the output at point C is +0.2v. This, in turn, cuts Q3 off and the output is +1.2v. But when both A and B are +0.2v, both Q1 and Q2 are cut off. Then the potential at point C begins to rise toward +6v. When it reaches +0.8v, Q3 begins to conduct and soon goes to saturation. Base or leakage current from Q3 flows through R4 and R3 which form a voltage divider. The base-to-emitter drop of Q3 plus the drop across R4 establish the level at point C at +1.2v. With Q3 conducting the output is +0.2v. Hence the output is +0.2v only when both inputs, A and B, are +0.2v.

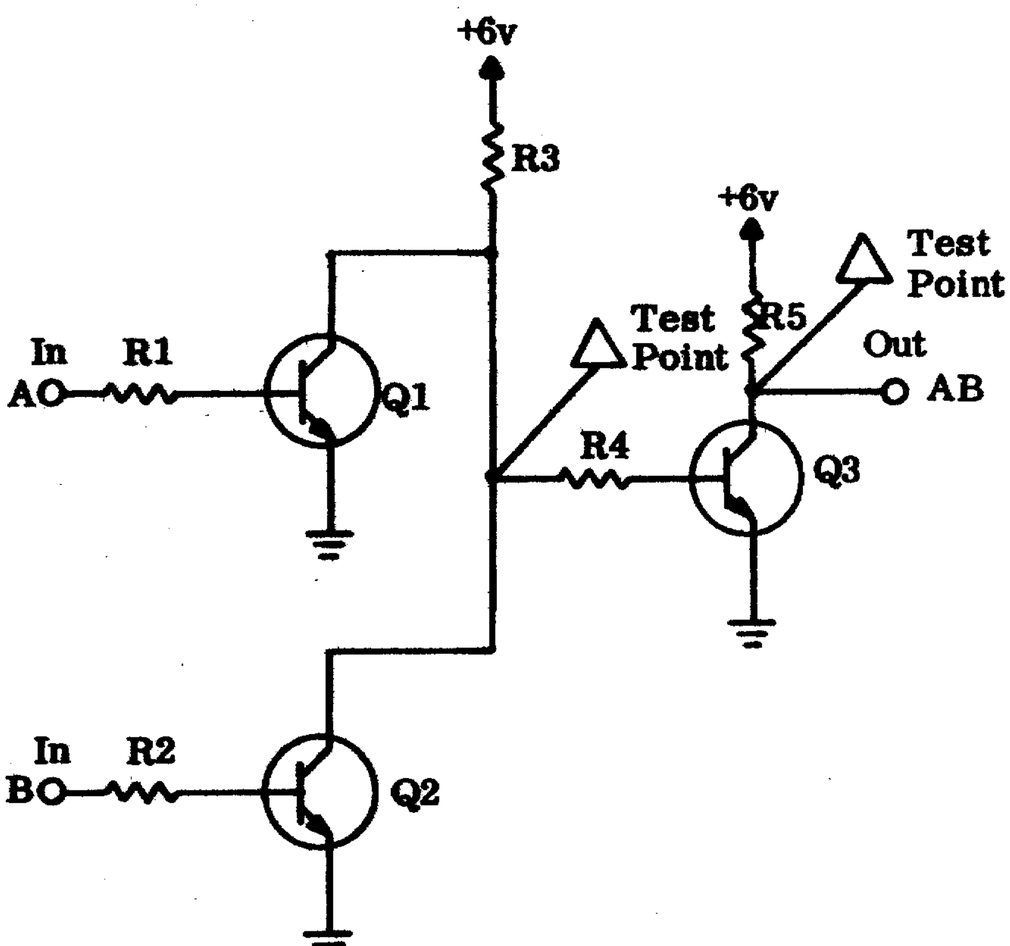
The circuit of figure 25 shows the OR combination of two signals, A and B, in which a +0.2v input at either A or B results in a +0.2v output.

LOGIC DIAGRAM SYMBOLS

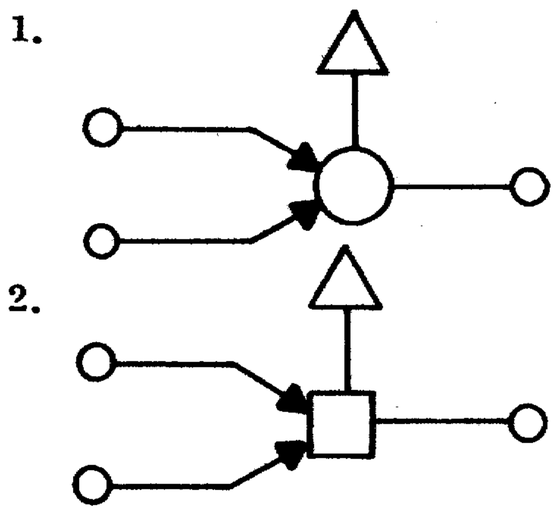
The symbols used to represent circuits on logic diagrams are shown in table 3. The use of two symbols, circle and square, to represent the combining of signals is a convenient way to indicate alternating logical levels. This alternation results from the inversion



Basic Combination (not AND)



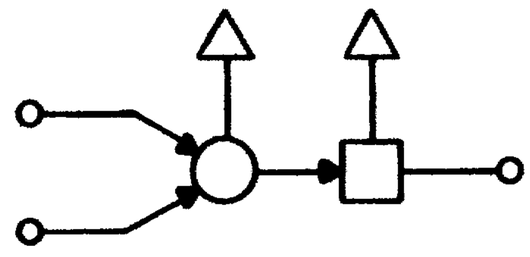
AND Combination



Logical Representation

Input A		Input B		Output	
+0.2v	1	+0.2v	1	+1.2v	0
+0.2v	1	+1.2v	0	+0.2v	1
+1.2v	0	+0.2v	1	+0.2v	1
+1.2v	0	+1.2v	0	+0.2v	1

Figure 23. Combining Signals



Logical Representation

Input A		Input B		Output	
+0.2v	1	+0.2v	1	+0.2v	1
+0.2v	1	+1.2v	0	+1.2v	0
+1.2v	0	+0.2v	1	+1.2v	0
+1.2v	0	+1.2v	0	+1.2v	0

Figure 24. AND Combination

Table 3. Circuit Symbols Used in Logic Diagrams

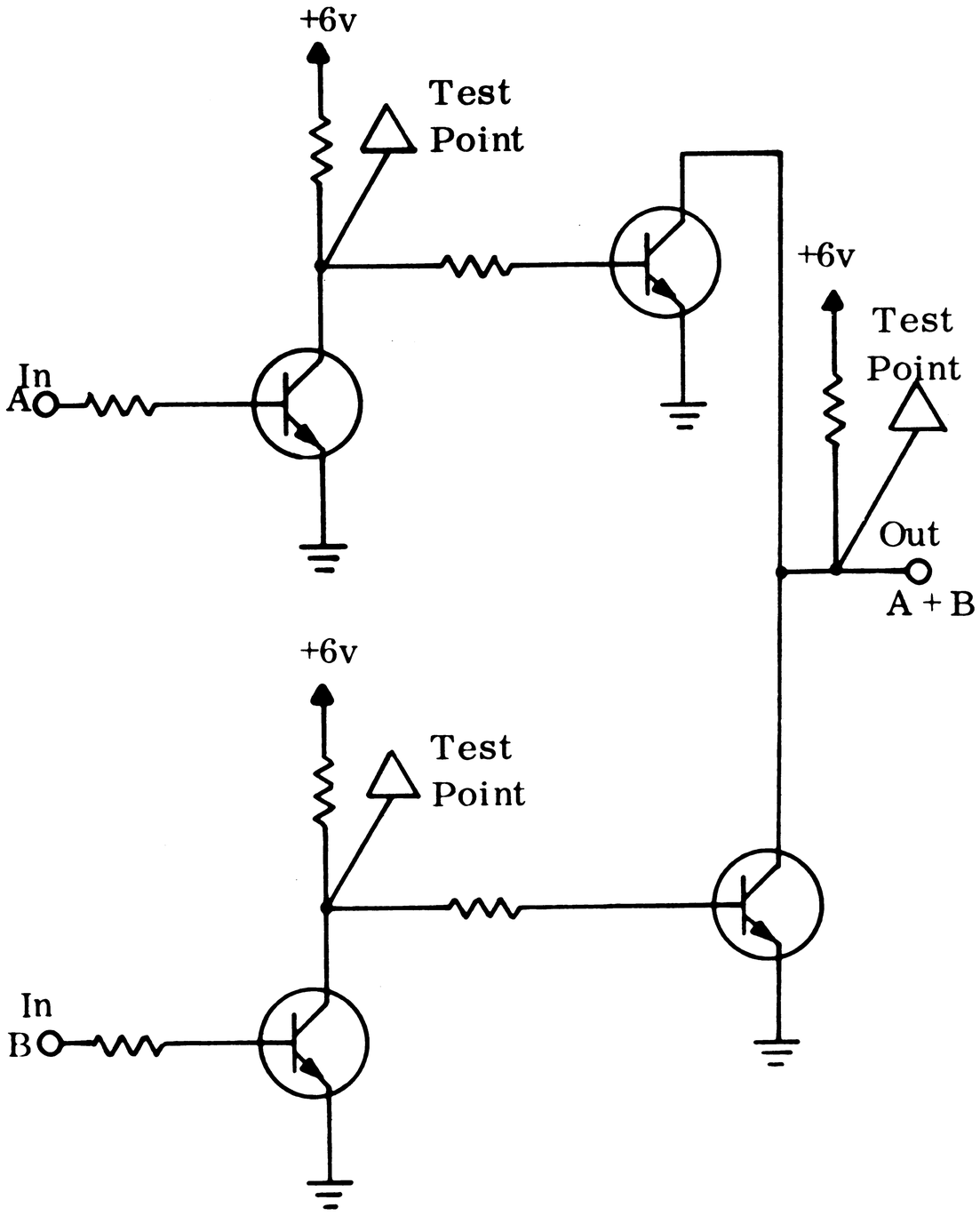
Symbol	Logical Function	Literal Significance
→	Inversion	Transistor (including its base resistor)
□	Usually OR combination (one or more zeros)	Collector load resistor
○	Usually AND combination (all ones)	Collector load resistor

accompanying each successive combination. Logic is performed by strings or successions of steps: invert, combine, invert, combine, invert, combine, etc. Thus, every other combining step is the reverse of its predecessor.

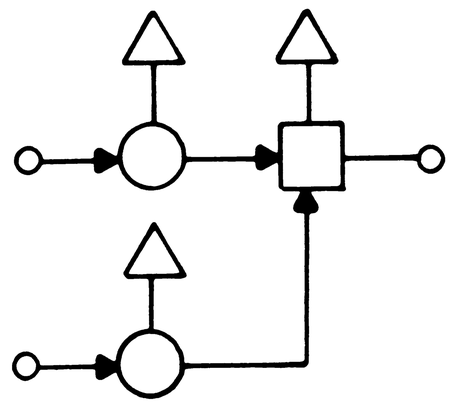
A diagram of such a logical string of steps appears in example 2.

Example 2





OR Combination



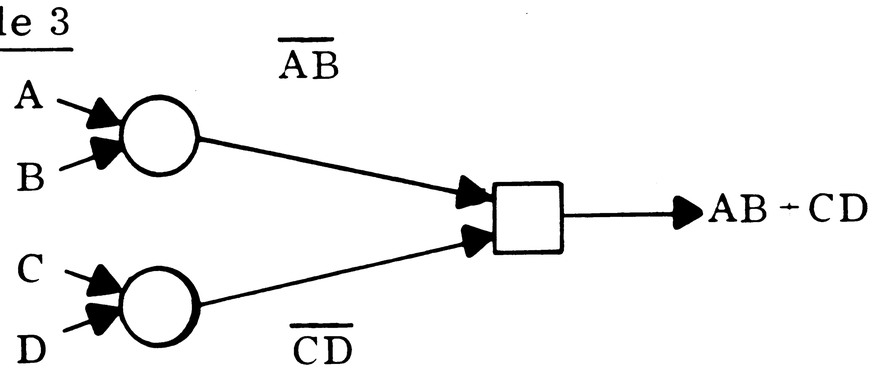
Logical Representation

Input A		Input B		Output	
+0.2v	1	+0.2v	1	+0.2v	1
+0.2v	1	+1.2v	0	+0.2v	1
+1.2v	0	+0.2v	1	+0.2v	1
+1.2v	0	+1.2v	0	+1.2v	0

Figure 25. OR Combination

Where possible, diagrams have circles and squares appearing in alternation and in such a way as to indicate whether the effect of a combination is an AND or an OR. Consider the logical circuit shown in example 3.

Example 3



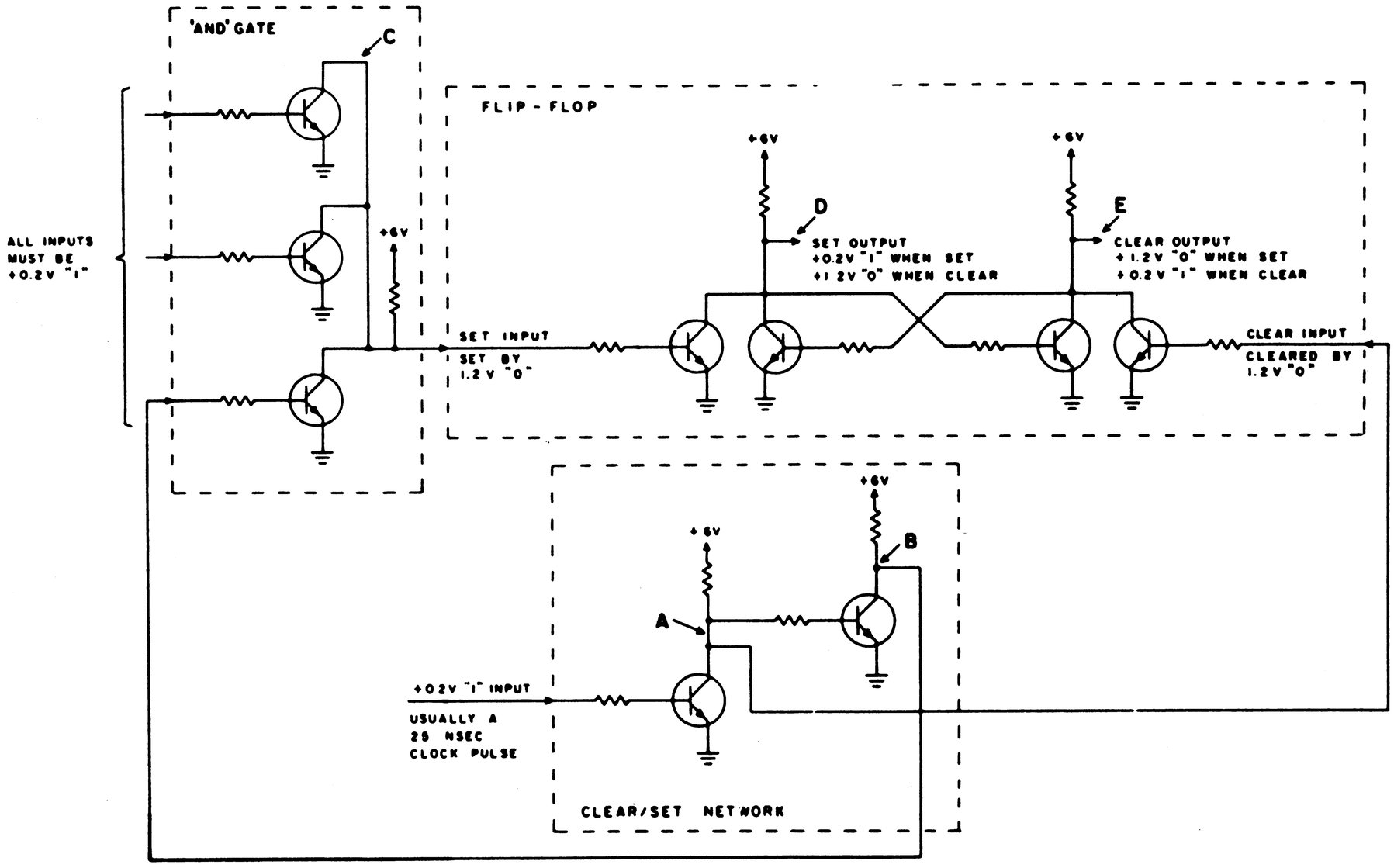


Figure 26. Basic Flip-Flop with Clear/Set Input

Its output is $\overline{\overline{A} + \overline{B}} + \overline{\overline{C} + \overline{D}}$ which simplifies to AB + CD. The square can be thought of as representing an OR.

FLIP-FLOP WITH CLEAR/SET INPUT

The circuit of a flip-flop and its clear/set input is shown in figure 26. The basic flip-flop can have up to five set inputs and five clear inputs. In this figure only one of each is shown. The set input is fed by a three-way AND gate, of which one input comes from the clear/set network.

The clear/set network enables the flip-flop to be cleared and reset by the same gating pulse. In many cases, this will be a 25ns clock pulse. Figure 27 shows the logical representation of this circuit and a timing chart for the clear/set operation.

A +0.2v one clear/set input to A results in +0.2v one outputs from B and E. This simultaneously clears the flip-flop while enabling AND gate C to reset it if one inputs are present. Assuming one inputs, the +1.2v zero output of C drives the output of D to a +0.2v one. In this state, D and E both are producing one outputs and the flip-flop is an indeterminate condition. However, when the clear/set pulse drops, the one output from A allows E to switch to zero, providing feedback to D which holds the flip-flop in the set condition. Meanwhile, the zero output of B has forced C to remove the set input from D, but the delay time through B and C allows the set input to stay up long enough after the clear input drops so that the flip-flop assumes the set state.

CIRCUIT LOADING

One circuit, which may be a common-collector connection of several transistors, can drive several other transistors which can be on the same module, or on a separate module, or a combination of the two. Figure 28 shows examples of loading.

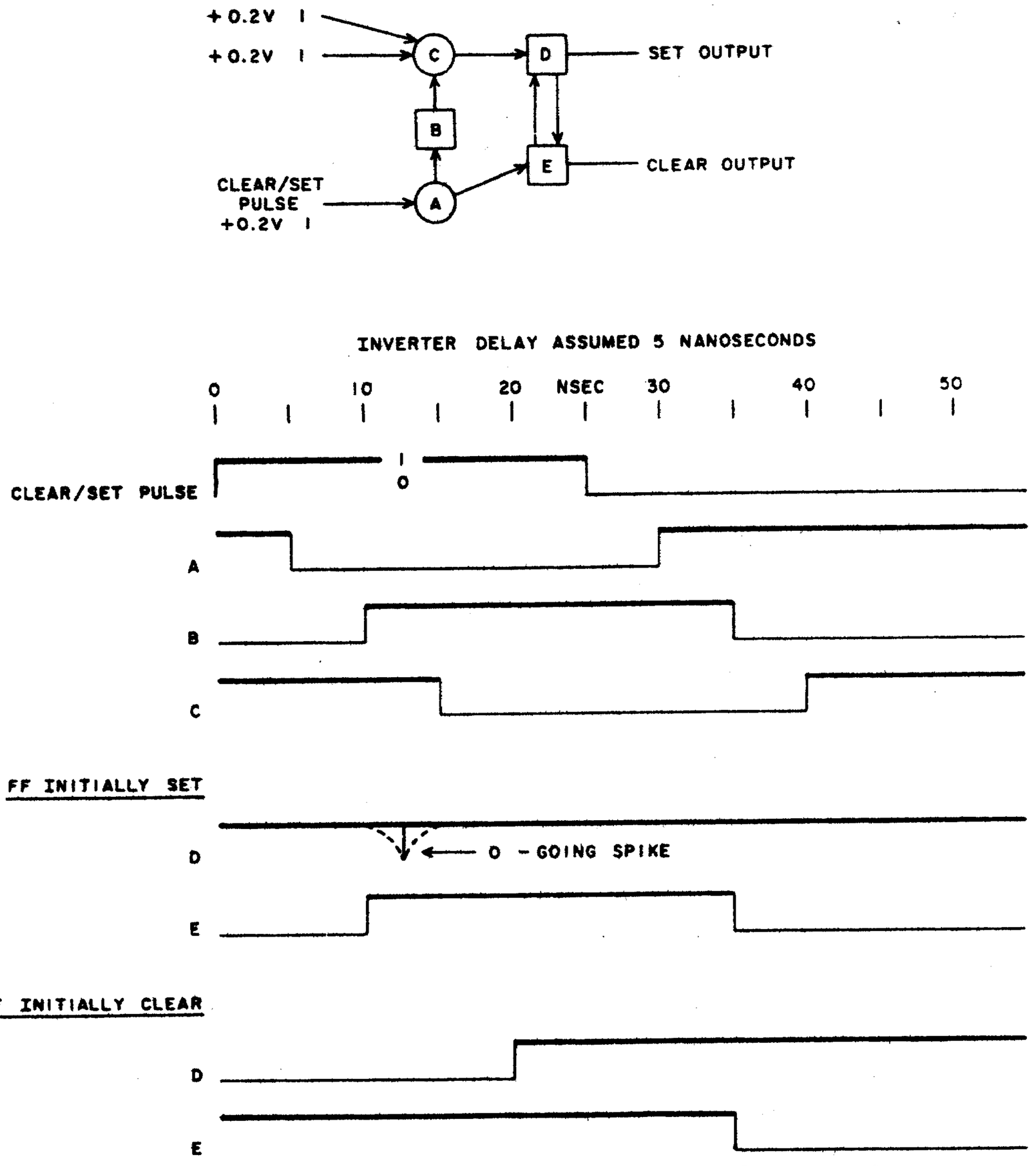


Figure 27. Flip-Flop Logical Representation and Timing Chart

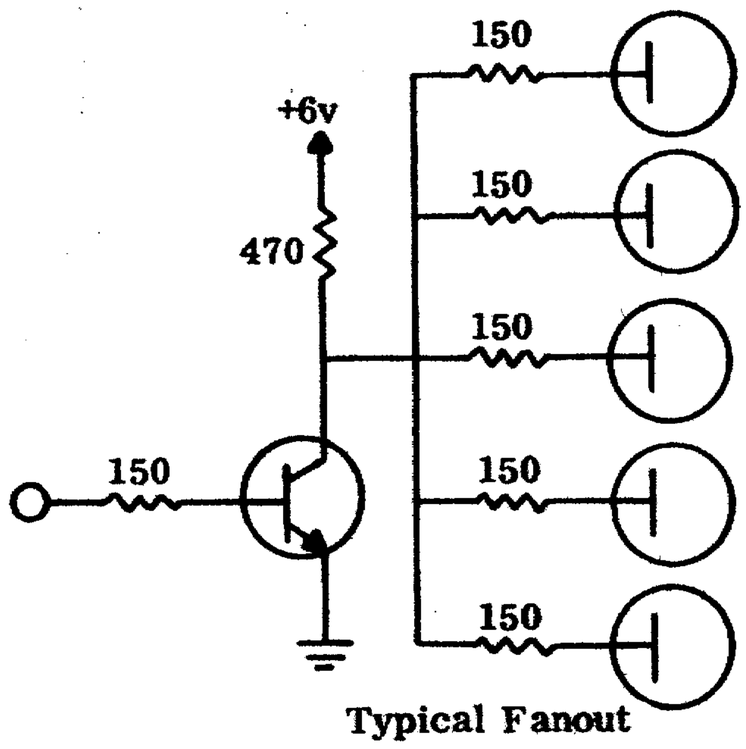


Figure 28. Base Resistor Loading

A circuit can drive up to five transistors when all transistors (including driver) are on the same module. This loading limitation is due to variations in base threshold voltage and corresponding base current demands. For each load configuration, the driver collector resistor and base resistors of the driven transistors are adjusted to balance and limit the base currents. This loading arrangement allows ample slave networks to provide multiple outputs.

Table 4 lists component variations for circuit loading within a module.

Table 4. Component Variations for Circuit Loading

	Driver Collector Resistor	Base Resistor of Driven Transistors
1 Load	680	47
2 Loads	680	100
3 Loads	680	150
4 Loads	560	150
5 Loads	470	150

Other load limits apply when the circuit load is on a separate module. A circuit can simultaneously drive up to two transistors on the same module and up to two transistors on a separate module. Loading in this case accounts for the loss on the transmission line connecting the two modules as well as variations in base threshold voltage noted earlier. The significance of this loading scheme is presented in the discussion on line drivers.

Table 5 lists component variations for a transistor driving loads within the same module, in addition to the twisted pair line.

Table 5. Component Variations

Number of Local Loads	Driver Collector Resistor	Output Line Resistor	Base Resistor of Driven Transistors	Output Shunt to Ground
None	470	56	---	120 plus Diode
1	470	56	100	---
2	470	56	200	---

LINE DRIVERS

Modules which are located on the same chassis are connected with twisted-pair lines; modules located on separate chassis are connected by coaxial cable. Line driver circuits are provided for each type of transmission line.

TWISTED-PAIR DRIVER

The twisted-pair driver is shown in figure 29. Initially, consider Q1 conducting so that its collector is at +0.2v. With Q2 off, the transmission line is terminated in an open circuit.

A +0.2v one input causes the collector of Q1 to

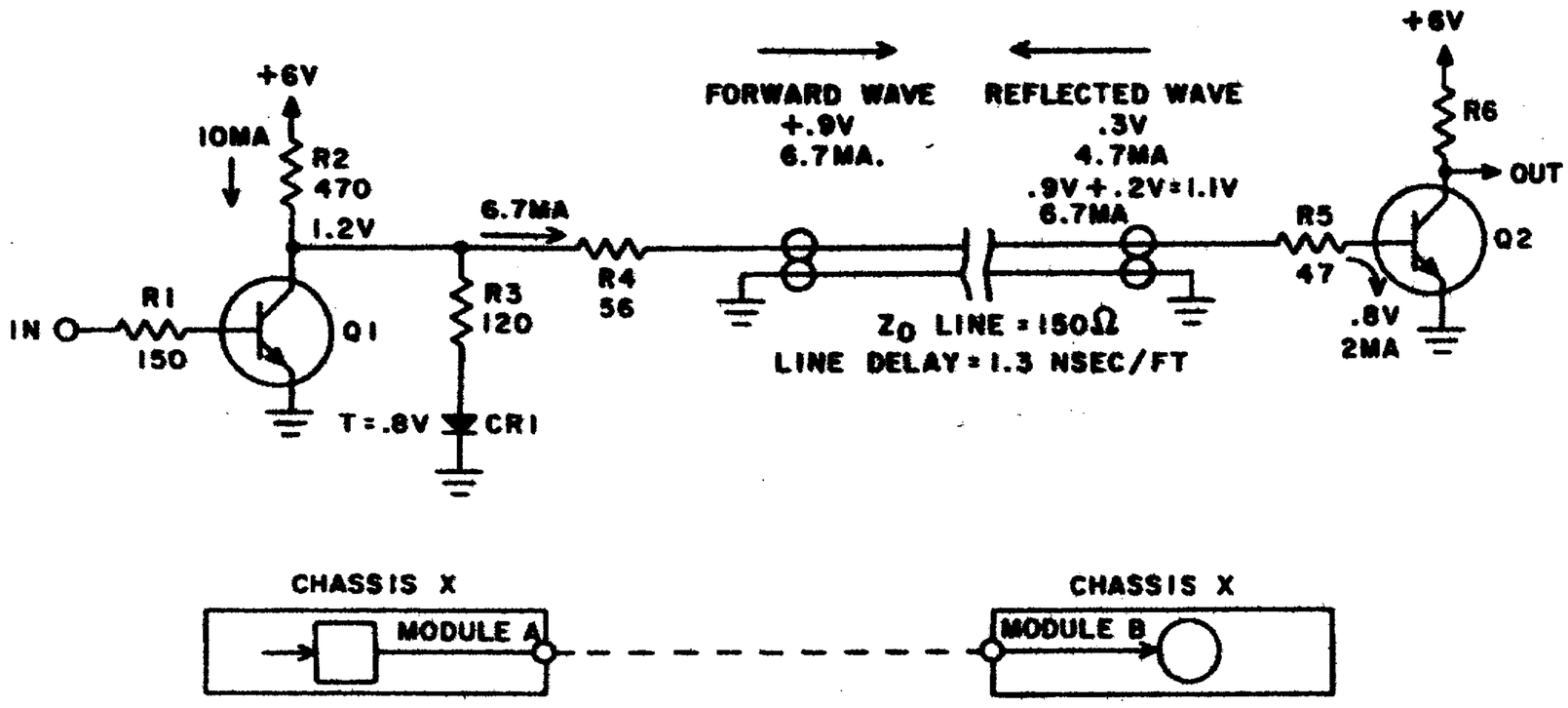


Figure 29. Twisted-Pair Driver

rise to +1.2v. This voltage is greater than the +0.8v diode threshold, and 3.3ma of the resulting 10ma current drawn through R2 is diverted through R3 and CR1 to ground. The remaining 6.7ma passes R4 and is placed on the line with a voltage of +0.9v (1.2v - 6.7ma x 56).

A finite time later this wavefront reaches the end of the line and appears at the input to R5, the base resistor for Q2. Since, at this instant, the voltage at the base of Q2 is +0.2v, this voltage adds to the incoming wavefront to result in a voltage of +1.1v at the input to R5.

At the next instant, the Q2 threshold is exceeded and base current flows. The base does not draw all of the current which is available in the line since it does not conduct appreciably below the +0.8v threshold. Thus, the source voltage for the base is +0.3v (+1.1v - +0.8v) and resistor R5 limits base current to 2ma.

Transmission line theory shows that power is reflected back to the source when the line is terminated in a load which does not absorb all of the transmitted power. Transmission line formulas for terminal voltage (V_T) and current (I_T) in this case show that the reflected wave has a current component of 4.7ma and a voltage component of +0.3v. The terminal point is at the output of R4, the input to the line.

Example 4

$V_T = V_F - V_R$ and $I_T = I_F - I_R$	Where: T = Terminal; F = Forward Component; R = Reflected Component
$V_R = +1.1 - (+0.8) = +0.3v$	
$I_R = 6.7ma - 2ma = 4.5ma$	

The reflected current is shunted to ground by diode CR1. In the absence of CR1, the line voltage rises to the +6v source.

The low base current, resulting from the line mismatch, retards the buildup of ground noise which occurs as a result of large ground currents.

The shunt diode and series resistor R3 are replaced by a transistor and its base resistor when it is necessary to drive a circuit on the same chassis and a circuit on another chassis. The transistor and diode thresholds are almost identical so that circuit operation follows the same principle.

The time required by a wavefront to travel down the line is a function of the type of line and its length. The twisted-pair line provides a theoretical delay of 1.3ns per foot. For example, the block system uses this feature to establish the 25ns duration of clock pulses.

The twisted-pair driver is represented on the logic diagrams by the standard square or circle. However, the output of the square or circle is connected to a pin of the module in question and wired from there to a pin on another module. The ground wire of the pair is wired to the connector ground bus of each module.

COAXIAL CABLE DRIVER

The coaxial cable driver is shown in figure 30. All transmissions between chassis are 25ns pulses on coaxial cables.

Initially, Q1 and Q2 are off so that their collectors are at the source voltage, +6v. No current flows in either primary winding. When Q1 is turned on, a 10ma current flows in the transformer primary winding to which its collector is connected. The voltage drop across R3 and step-down action of the transformer (and transformer loss) produce a voltage pulse on the secondary winding of approximately +1.38v.

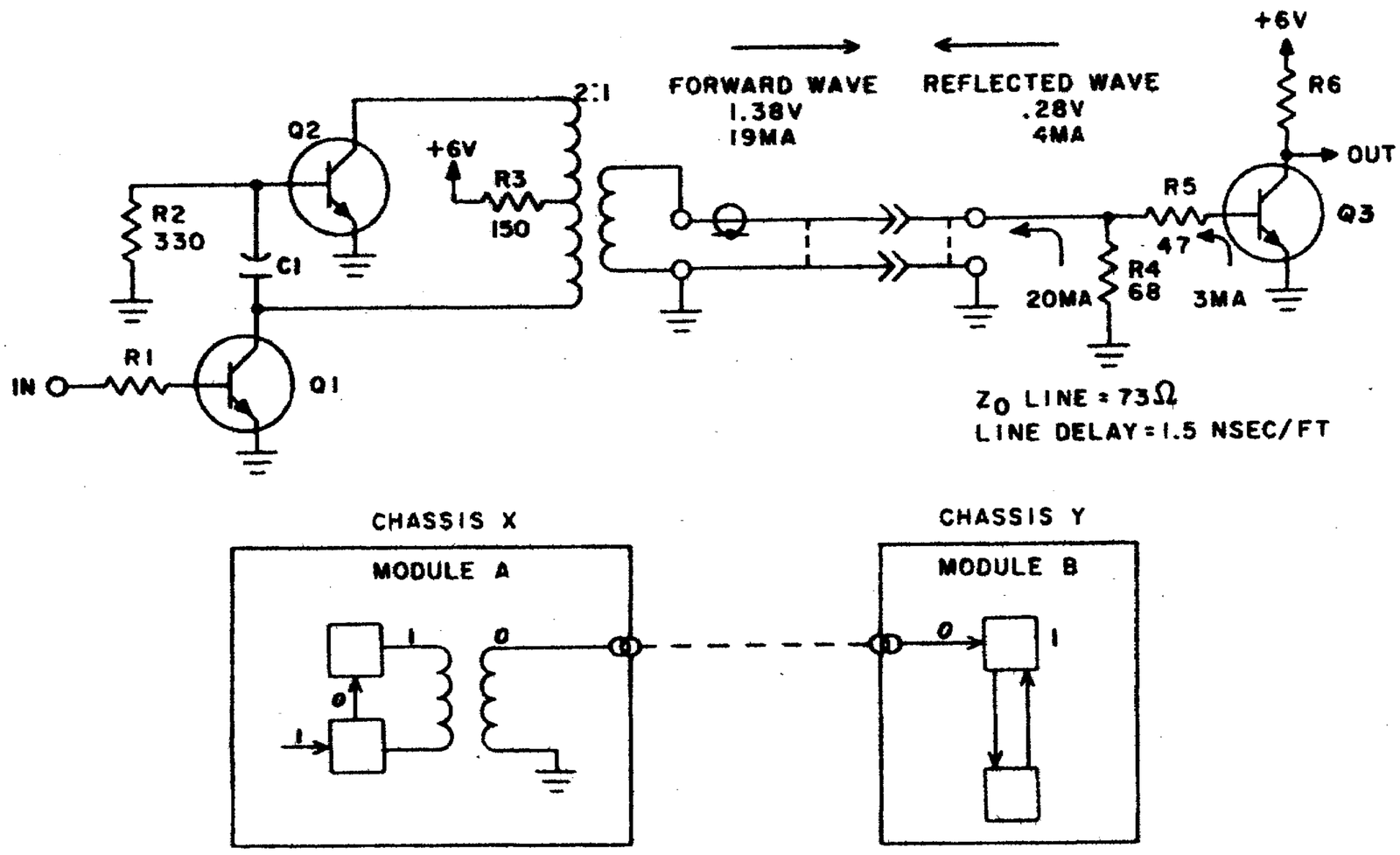


Figure 30. Coaxial-Cable Driver

The pulse duration on the line follows the 25ns pulse on the base of Q1. This pulse travels down the coaxial cable and turns transistor Q3 on. Resistor R4 terminates the line in slightly lower than its characteristic impedance of 75 ohms. Approximately 20ma of secondary current is shunted to ground through R4. Q3 draws approximately 3ma base current.

Capacitor C1 couples the change in Q1 collector voltage to the base of Q2. Capacitor C1, R3, R2, and the base/emitter diode of Q1 cause Q2 to conduct about 25ns later than Q1. When Q2 conducts, current flows in the primary winding to which the Q2 collector is connected. The resulting signal in the transformer secondary is equal to but of opposite polarity from the previous signal and lags by 25ns. The net effect of this is to provide an equal and opposite magnetizing force for each pulse in the transformer. This prevents the transformer from building up a bias on the line which could occur under the high switching rates.

The coaxial cable driver circuit is represented on logic diagrams as shown in figure 30. The time delay on the 75-ohm coaxial cables is 1.5ns per foot. All coaxial cable lengths are standardized at 10 feet to allow synchronous timing signals on all chassis. Coaxial cables pass directly through a chassis feed-through connector (without soldering) and connect directly to the transmitting or terminating circuit.

CHAPTER 3

PERIPHERAL AND CONTROL PROCESSORS

INTRODUCTION

The Control Data 6000 Series computers have 10 peripheral and control processors. Each peripheral and control processor is an independent computer with 4096 words of core storage and a repertoire of 64 instructions. The peripheral and control processors share access to central memory (read and write) and 12 bidirectional I/O channels.

The 10 processors are combined in a multiplexing arrangement which allows them to share common hardware for arithmetic, logical, I/O, and other operations without sacrificing speed or independence. This multiplexing arrangement consists of a barrel and slot and common paths to and from storage and I/O circuits (figure 31).

The barrel is a matrix of flip-flops used to hold the current instruction and operand for each of the 10 processors and to give each a turn to use the execution hardware in the slot (adders, shift network, etc.). The quantities in the barrel are shifted from slot output to slot input (counterclockwise in figure 31), and each time a processor's data enters the slot a portion of the instruction is executed (one trip around the barrel requires 1000 ns, or one major cycle).

The slot is the execution hardware (adders, incrementors, etc.) which does the actual arithmetic and logical operations, advances the program address, etc.

Each processor has its own independent 4096 word memory which may be referenced once each major cycle (once each trip around the barrel).

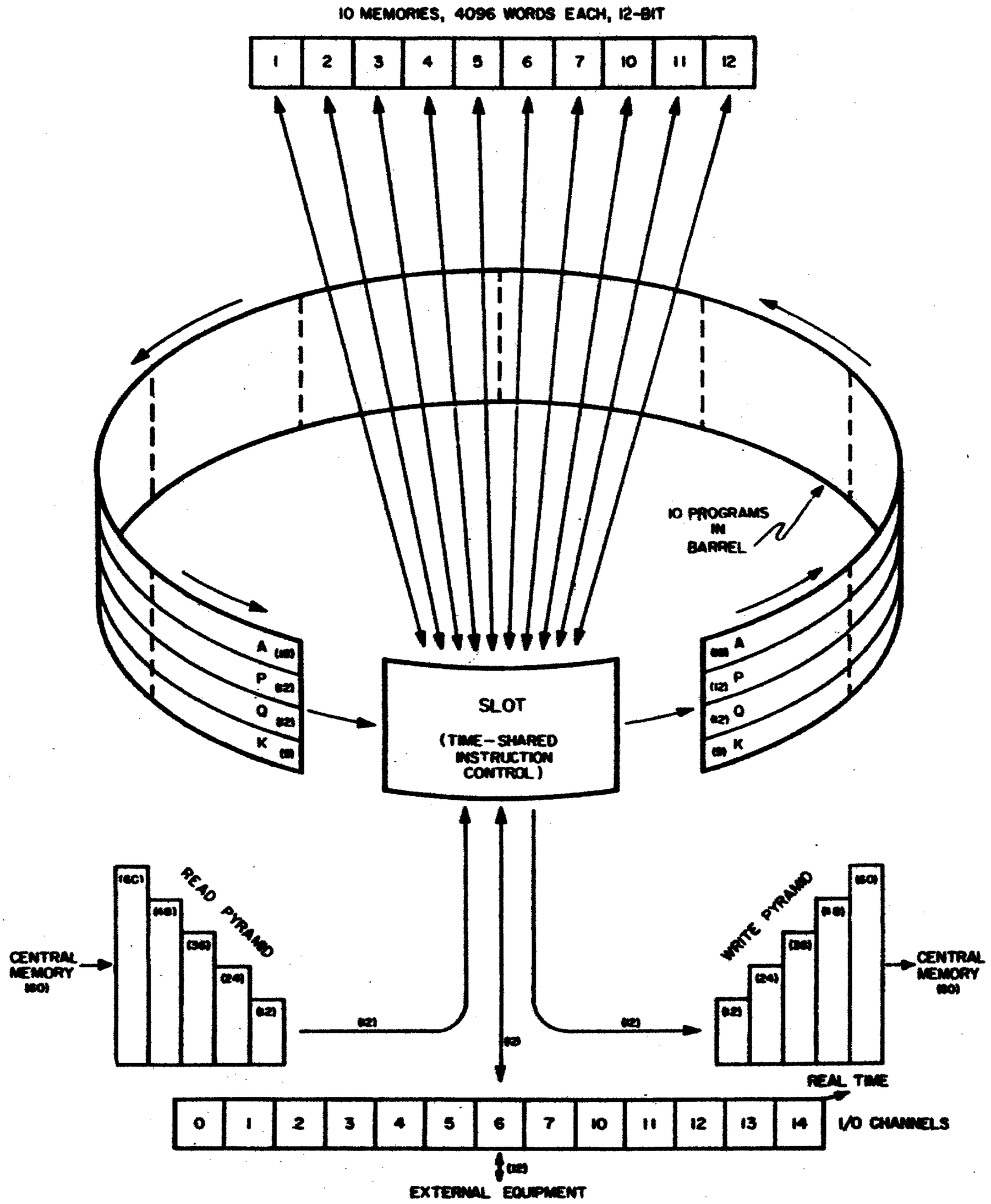


Figure 31. Peripheral Processors

There are 12 bidirectional I/O channels which are available to all 10 processors.

The peripheral and control processors read data from input devices, perform preliminary arithmetic and logical operations, send data and programs to central memory, assign tasks to the central processor, read central processor results from central memory, and send results to external storage (magnetic tape, disk file, etc.) or output devices (line printer, display console, etc.).

PERIPHERAL AND CONTROL PROCESSOR CHARACTERISTICS

Characteristics of the peripheral and control processors are:

- 4096 word magnetic core storage (12-bits), random access, coincident current. Major cycle is 1000 ns. Minor cycle is 100 ns.
- Twelve bidirectional I/O channels which are available to all processors. Maximum transfer rate per channel is one word/major cycle.
- Real-time clock uses 4096 counts per clock period (major cycle).
- The instructions set includes arithmetic, logical, I/O, central memory read/write, and exchange jump instructions. Average instruction execution time is two major cycles.
- Indirect addressing and indexed addressing are available.

A peripheral and control processor instruction may have a 12-bit or a 24-bit format. The 12-bit format has a six-bit operation code (F) and a six-bit operand or operand address (d). The 24-bit format uses the 12-bit quantity (m), which is the contents of the next program address (P + 1), with d to form an 18-bit operand or operand address.

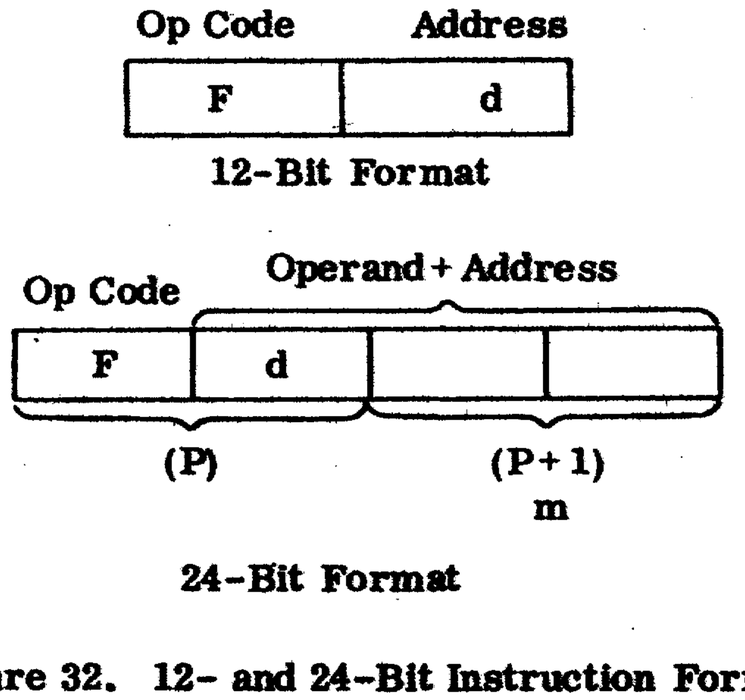


Figure 32. 12- and 24-Bit Instruction Formats

The two instruction formats allow several modes of addressing operands which include: no addressing, direct addressing, indirect addressing, memory addressing, and constant addressing. See 6000 Series Reference Manual for explanation of each mode.

TIMING

Timing is controlled by a master clock located on chassis 8 in the 6400 and on chassis 1 in the 6600. Timing of the peripheral and control processors is controlled by a four-phase clock regeneration system (figure 33). Four 25-ns pulses are issued each minor cycle to control movement of data and instructions. A storage sequence control system, timed by the four-phase clock, controls storage references and defines the 10 peripheral and control processors.

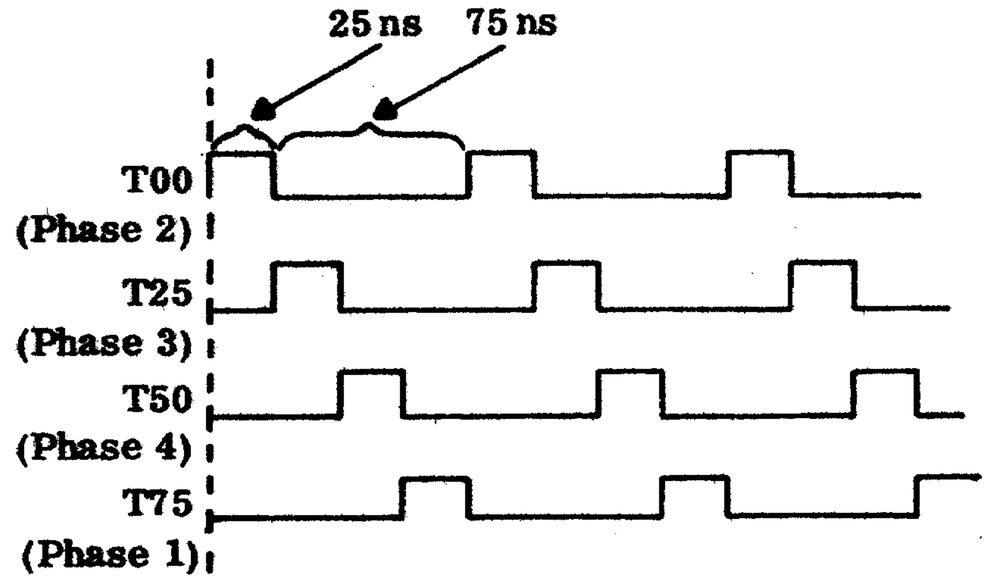


Figure 33. Four-Phase Clock Regeneration System

PPU TIMING

The timing for the peripheral processor chassis is a result of the clock regeneration circuit located on chassis 1 (figure 34). A pulse is received on chassis 1 from the master clock on chassis 3 every 100 ns. The clock regeneration circuit located on chassis 1 regenerates four 25-ns pulses which are, in turn, fanned out for timing.

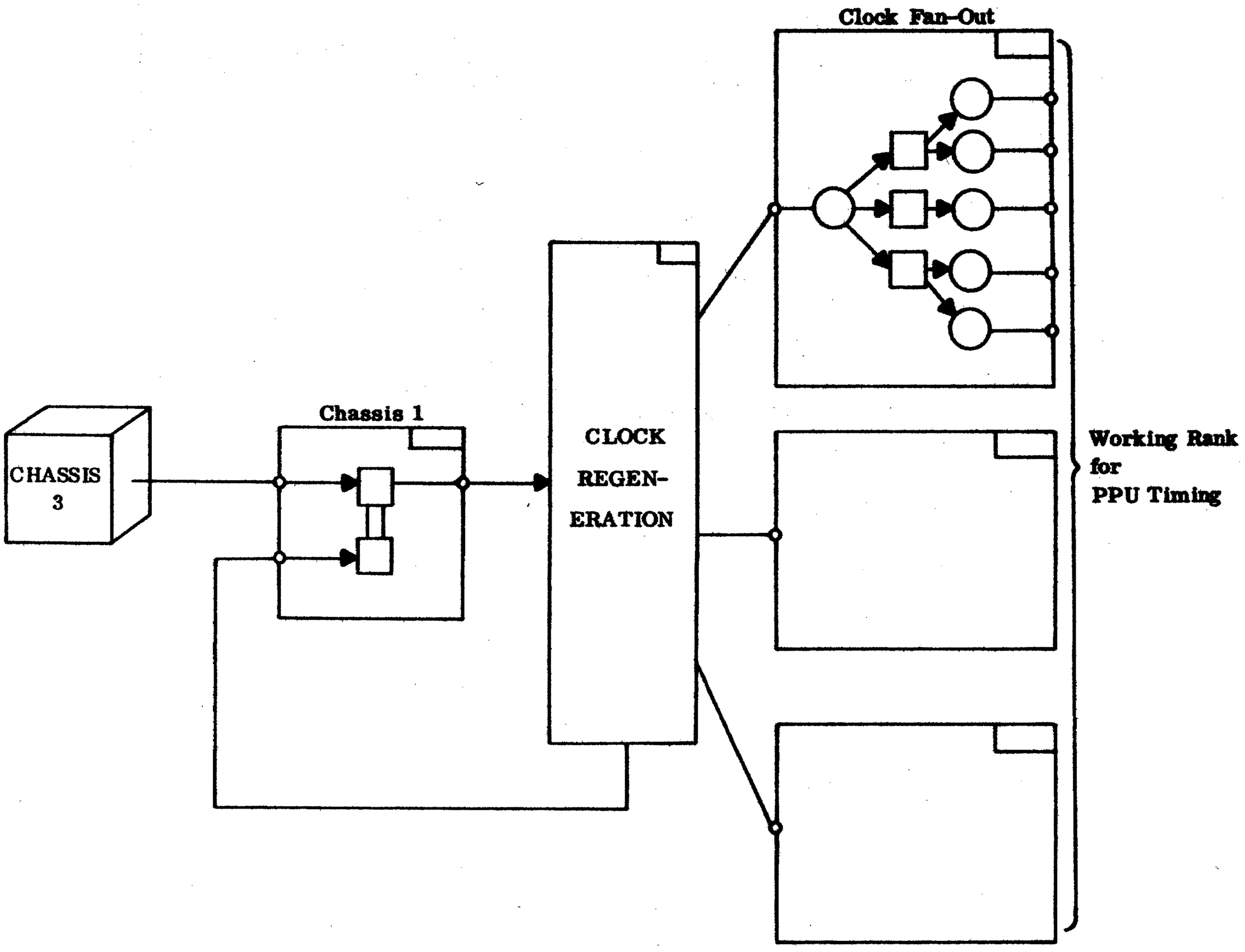


Figure 34. Clock Regeneration Circuit

REGISTERS

The barrel contains the A, P, Q, and K registers for each of the 10 processors. The functions of these four registers in the barrel are as follows:

- A register (18 bits) holds one operand for add shift, logical, and selective operations. The 18-bit quantity in A may be an arithmetic operand, central memory address, or an I/O function or data word.
- P register (12 bits) is the Program Address register. (P) is also used as a data address in certain I/O and central instructions.
- Q register (12 bits) holds data for a variety of functions. These include: The address of the operand during direct addressing; The address of the address of the operand on indirect addressing; The peripheral address of data used during central read or write instructions; The upper six bits during constant mode instructions; The channel number on all I/O and channel jump instructions; The shift count on shift instructions; The specific number of locations to jump on relative jumps.
- K register (9 bits) holds the F portion of an instruction word and the trip count (the number of times an instruction has been around the barrel).

Figure 35 shows the inputs and outputs of the registers in the barrel and the sequence in which they occur.

A storage address for a particular processor is taken from the barrel six minor cycles before that processor is ready to enter the slot. This allows time for operands to be obtained from storage and entered into the barrel before slot time. Similarly, K is sent to translation networks in time to be fully translated

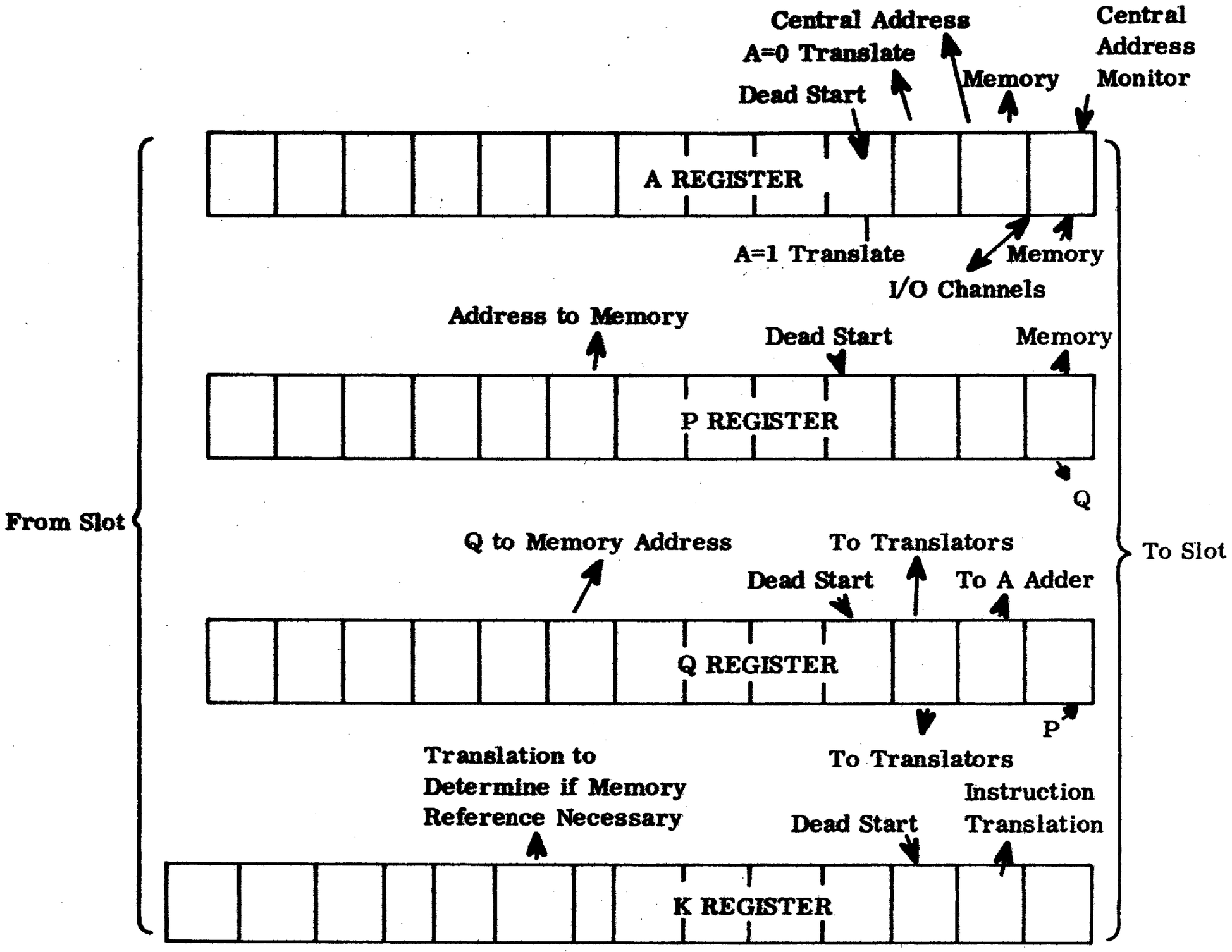


Figure 35. Barrel

before the processor reaches the slot. The dead start inputs shown in figure 35 are discussed in the section on dead start.

A REGISTER

The A register in the barrel receives the results of add, shift, logical, or selective operations in the slot. Each result may be stored, returned to the slot unaltered, or used to condition other operations. The A register tests to determine whether it is positive, negative, zero, nonzero, or one. The results of these tests can be used to condition jumps or other instructions. The quantity in A may be a full 18 bits, as in the case of central instructions, or the A register may contain a 12-bit peripheral word in which case the upper bits are zero.

The connections to A in the barrel are described in the following paragraphs.

Outputs

A ⟶ M. (A) can be sent as a data or function word on one of the I/O channels.

A ⟶ Central Address Register. (A) is the central memory address in Central Read and Write and Exchange Jump instructions.

A ⟶ Y. This is a Store instruction. (A) is sent to Y and then to storage.

A ⟶ Translation Networks. This is for sign and magnitude checks.

Inputs

X ⟶ A. The content of the central Program Address register is sent to the peripheral X register every minor cycle. A 27 instruction sends (X) to A and enables a peripheral and control processor to monitor the progress of the central program.

R⟶A. An input to A instruction gates a word from an I/O channel into A.

Fd⟶A. A data word from storage is entered into A by the Fd⟶A path.

A⟶A. When the quantity in A is to be returned to the slot unaltered, the A⟶A gate is enabled.

10,000⟶A. When dead start is initiated, 10,000 ⟶ A.

P REGISTER

P holds the program address and is not changed in the barrel except in the case of dead start. P is incremented in the slot. As shown in figure 35 (P) is sent to a storage unit before slot time. This allows time to read a word from storage and make it available at slot time. (P) is sent to the G register which feeds all 10 storage address or S registers (figure 36).

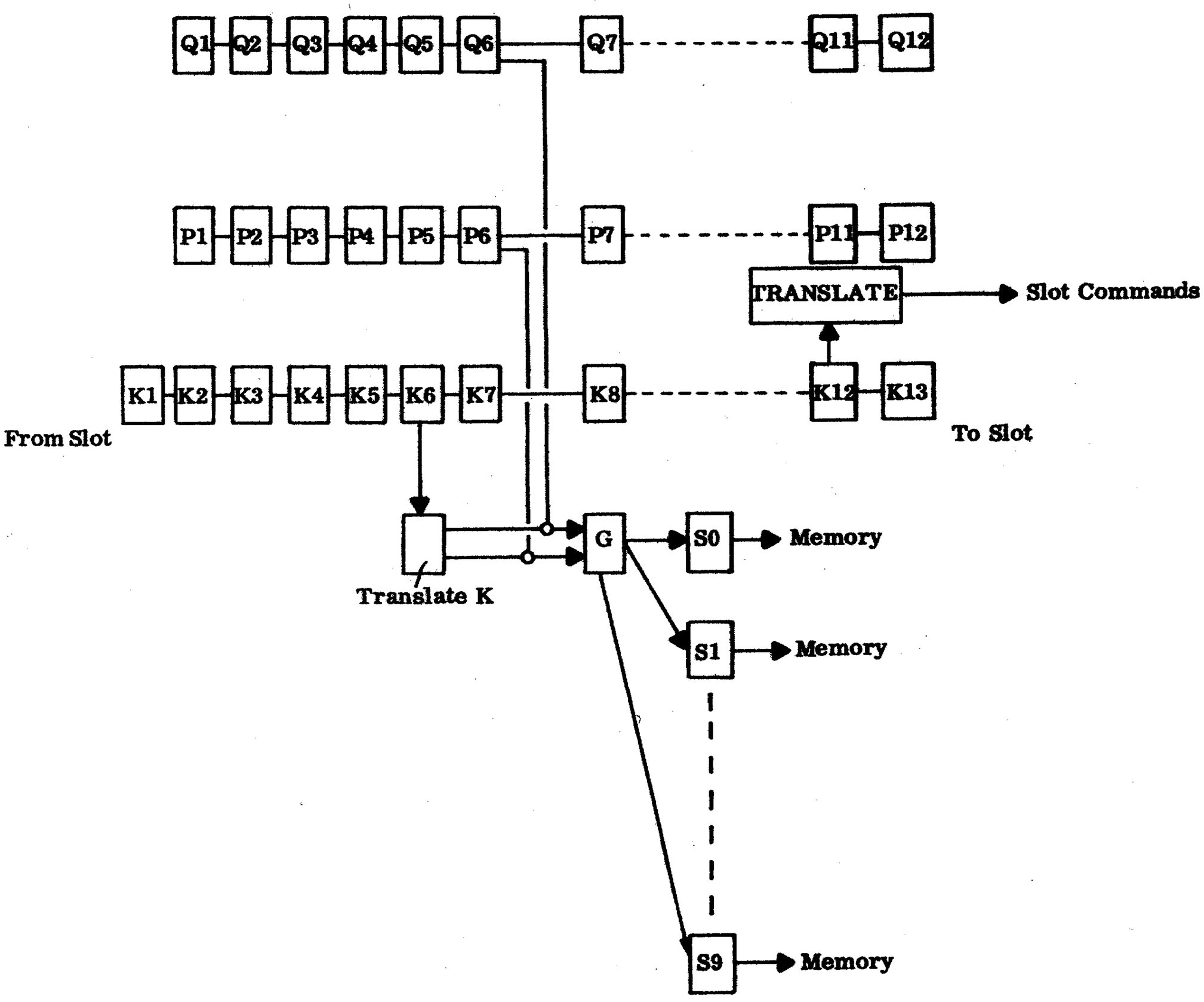


Figure 36. (P)⟶G Register⟶S Registers

When a jump is called for, P is sent to Q. Q is then altered by the Q adder in the slot and the new address returned to P at the first stage of the barrel.

Q REGISTER

Q holds the d portion of an instruction and has several outputs to translation networks which select channels during I/O instructions. When d is an address (direct mode of addressing), d is sent from the slot to Q in the barrel and the word obtained from that address is entered into A in the slot. When a jump is called for, the quantity in Q is added to or subtracted from (P) in the Q adder and the result sent to P. When an instruction calls for an 18-bit operand, the lower six bits of Q are sent to the upper six bits of A to form the 18-bit quantity dm.

K REGISTER

K holds the F portion of an instruction word and a three-bit trip count which sequences the execution of

an instruction. K is translated at two different times during a trip around the barrel: first, to determine if a storage reference is needed; second, to provide the proper commands at the slot (figure 36). During the barrel trip in which a new instruction is being read from storage, a translation of K = 00X enables translations from Fd in the storage cycle path to be used in place of K translations. This eliminates the need for a separate RNI trip through the barrel and allows some instructions to be read from storage and executed all in one trip. The K = 00X translation arises from the fact that K is cleared at the end of each instruction.

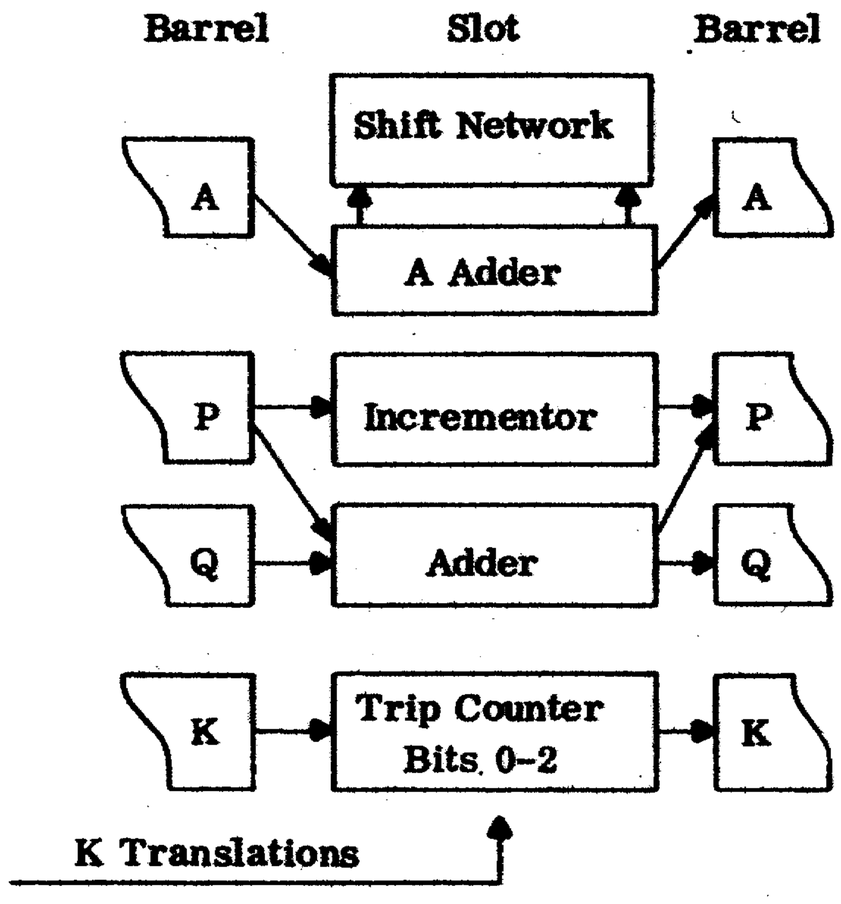


Figure 37. Elements of the Slot

SLOT

The slot contains the execution hardware for A, P, Q, and K, and each processor is allowed one minor cycle in the slot every major cycle. Included in the slot are the A and Q adders and the P and K incrementors.

As A, P, Q, and K enter the slot (figure 37), K translations (started earlier in the barrel) become available and a portion (or all) of an instruction is executed. The results are gated back into the barrel to be stored, used again, or sent to I/O equipment.

A ADDER

The A adder (figure 38) is used to execute Add, Subtract, Selective Clear, Logical Product, and Logical Difference instructions. Parts of the A adder are also used to enter a word into the shift network and gate the result back to the barrel. The quantity in A in the barrel is always complemented when it enters the slot. When no operation on A is called for, (A) is complemented, sent to the A adder, and added to zero. The result is complemented at the adder output. Since the A and Q adders are basically alike, see the Q adder for detailed description of the adder operation. The add gate on the QD modules (figure 38) is always enabled, except when Selective Clear, Logical Product, or Shift instructions are enabled.

Add

For an Add instruction, (A) is complemented and entered into the A Input register. The second operand is also inverted before entering the B Input register. The two quantities in the input registers, taken as positive, are added and the sum is complemented as it is gated out of the QD modules to the barrel.

Subtract

For Subtract instructions, the minuend, in the A register, is complemented in the adder. The subtrahend is entered into B without being complemented and the two quantities are added as in an Add instruction.

Selective Clear

For selective clear operations, the complement of A and the true value of d are entered into the adder while both the selective and the logical product gates are enabled.

Logical Product

For Logical Product instructions, both A and d (or dm) are complemented before entering the adder and both the logical product and the selective gates are enabled.

Logical Difference

For Logical Difference instructions, the complement of A and the true value of the second operand enter the adder, and only the selective gate is enabled.

Shift Network

The Shift instruction (10) provides for shifting the number in A up to 31 places left or right. Left shift is circular with the high-order bits reentering A at the low-order end. Right shift is end-off with low-order bits discarded as they shift out of the A register with no sign extension. Thus, a left shift of 18 is equivalent to no shift, and a right shift of 18 clears the A register.

The shift network (figure 39) is a static network. The content of A enters the register at time 50, each bit follows a path established by static translations of the six-bit shift count in d, and the result reenters A in the barrel at the next time 50. The input to the shift network comes from the A Input register in the A

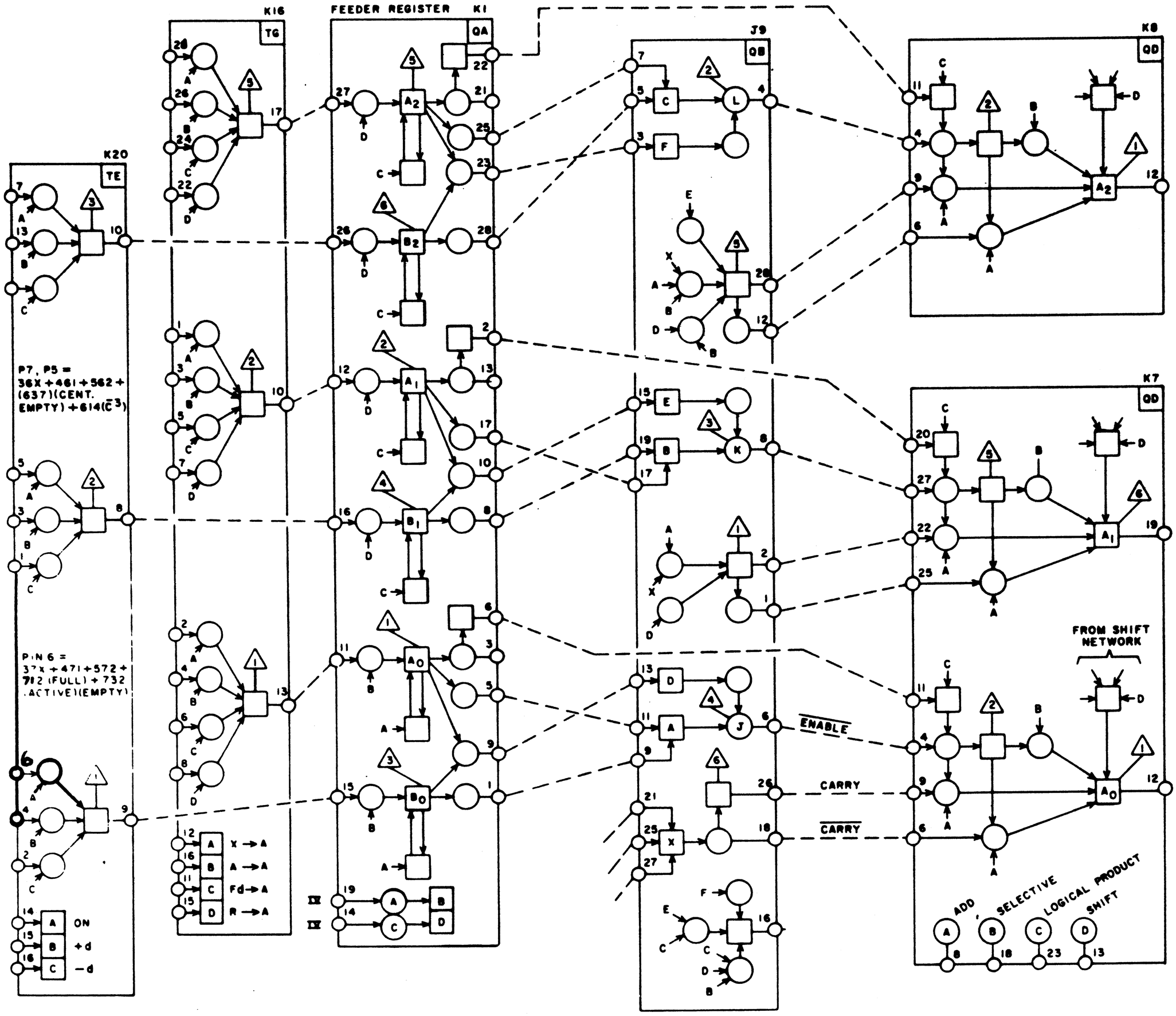


Figure 38. A Adder

adder (the content of that register, $\overline{A}$, is complemented before entering the Shift register). The output of the shift network is gated back to the barrel by way of the output modules (QD) of the A adder. Note that the quantity in A is always shifted, but the result is gated to the barrel only when the current instruction is a Shift instruction.

If d is positive ($00-37_8$), the shift is left and the shift count is the content of d. If d is negative ($40-77_8$), the shift is right and the shift count is the complement of the number in d.

At the first stage of the shift network, d4 and d5 are tested to determine whether the shift is greater or less than 16 and whether it is right or left (figure 40). If the shift is 16 or greater, a shift of 16 is made at this point, and the result then enters the rest of the shift network.

Bits d0 through d3 are tested with d5 as shown in figure 41 to set up paths through the rest of the network. Figure 39 shows the path of bit 0 for a left shift of six places.

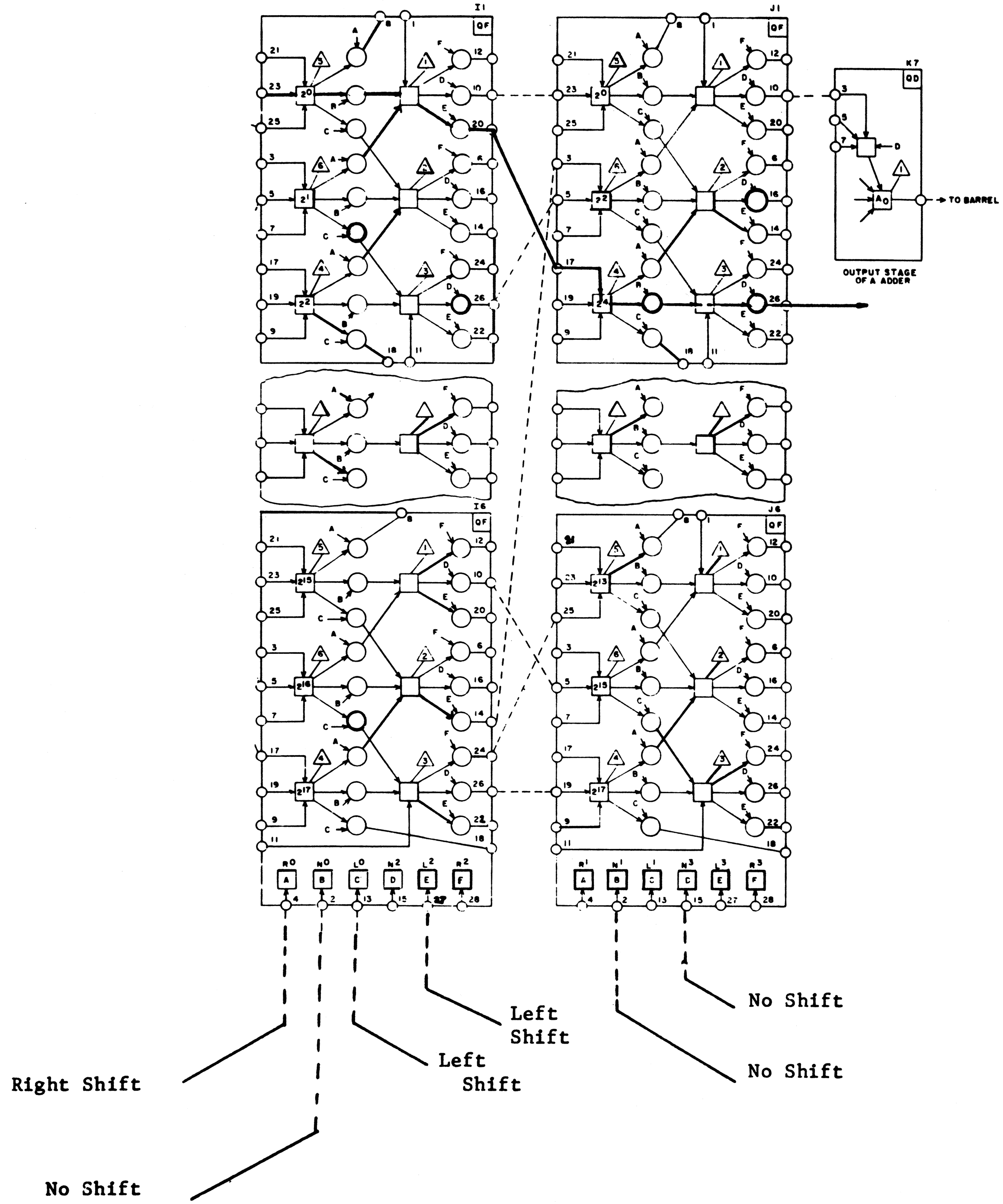


Figure 39. Shift Network

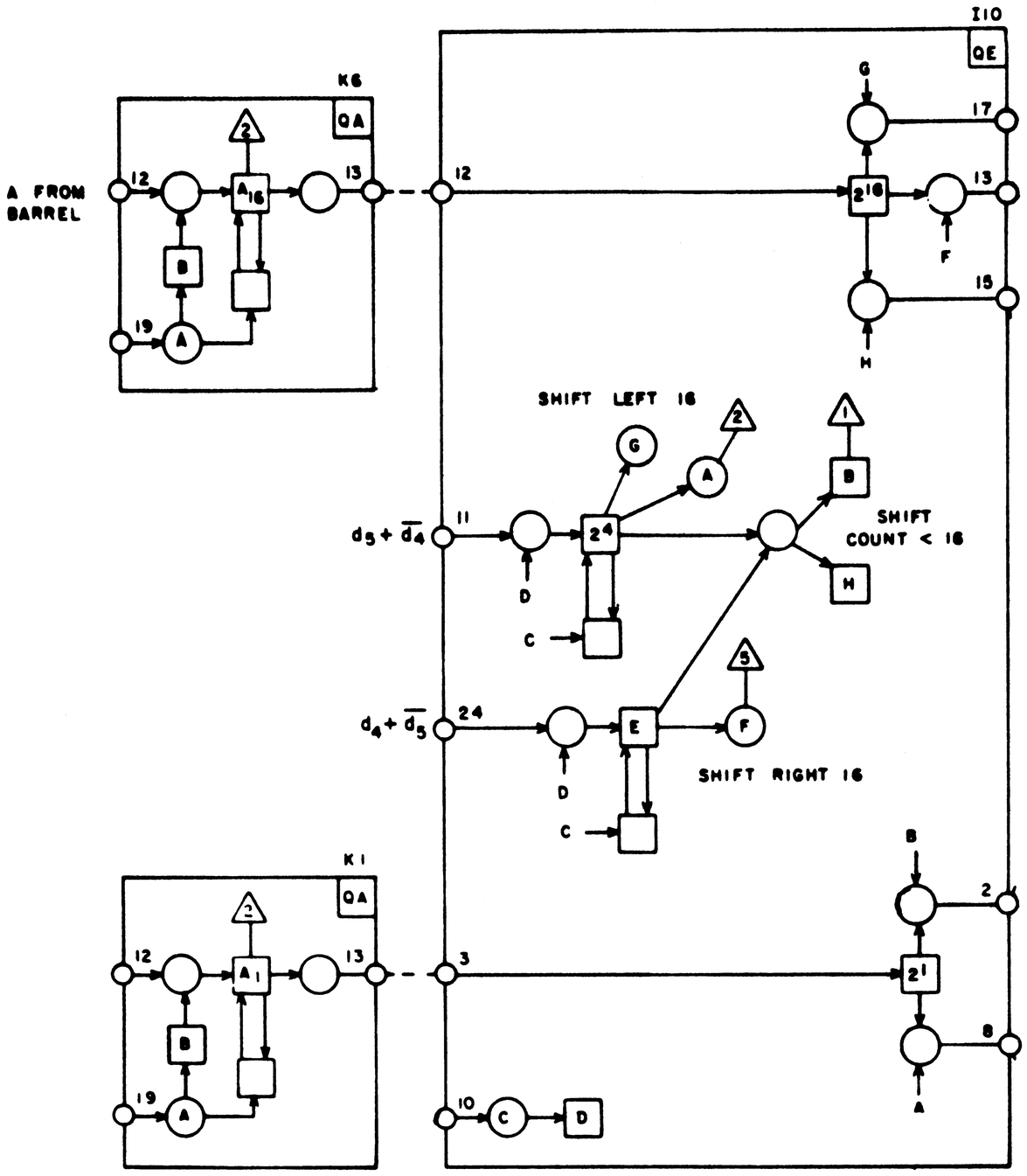


Figure 40. Shift

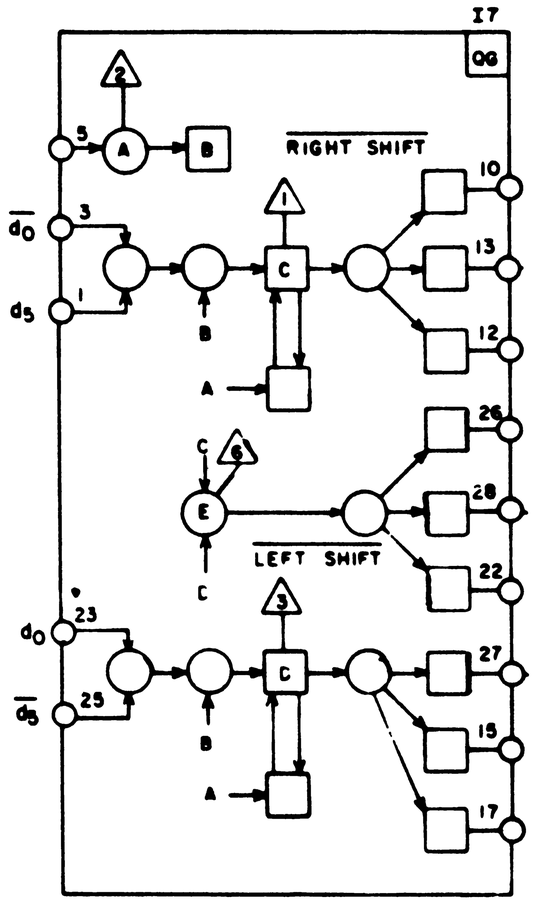


Figure 41. Shift Path Control

P INCREMENTOR

The P register in the slot is an incrementor (figure 42) which can advance (P) by one or send (P) back to the barrel unchanged. When P is to be changed by more than one, the arithmetic is done in the Q adder and the result is entered into the first FF of P in the barrel.

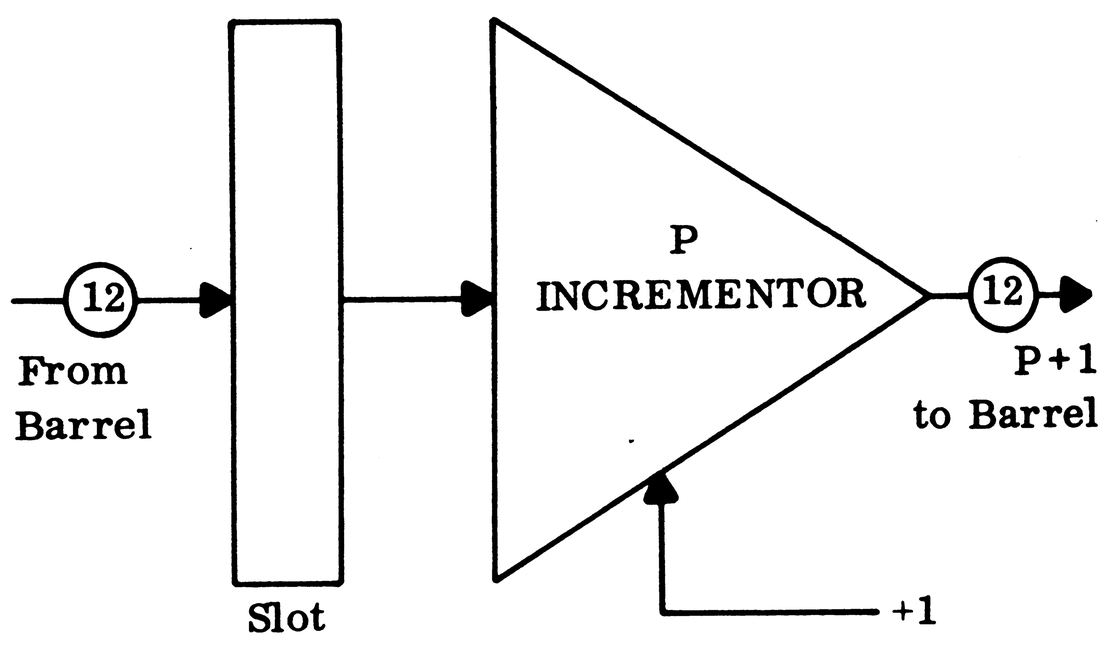


Figure 42. P Incrementor

Q REGISTER

Q in the slot is a 12-bit adder (figure 43) which adds and subtracts in the same way as the A adder. For Jump instructions, P is sent to Q before the slot. The content of Q (now the program address) is complemented and sent to the Q Input register and the complement of d is sent to the lower six bits of the H Input register. If d is positive ($01\text{-}37_8$), the upper six bits of H are set to 77 by the 00 ⟶ H gate. If d is negative ($40\text{-}77_8$), the upper six bits of H are set to 00. No input to H gate is enabled.

GENERAL THEORY OF ADDERS

In general, adders are usually defined in two manners: whether they are one's or two's complement; or, whether they are additive or subtractive. A one's complement adder is one where an end-around carry is used. Subtractive or additive definition is based on the result that the adder will yield for certain sets of operands. In an additive adder, positive zero as a result is only obtainable by adding positive zero to positive zero. All other combinations of adding ± zero and ± zero will result in negative zero.

Example 5

+0	-0	+0	-0
++0	++0	+-0	+-0
+0	-0	-0	-0

Additive Adder Results

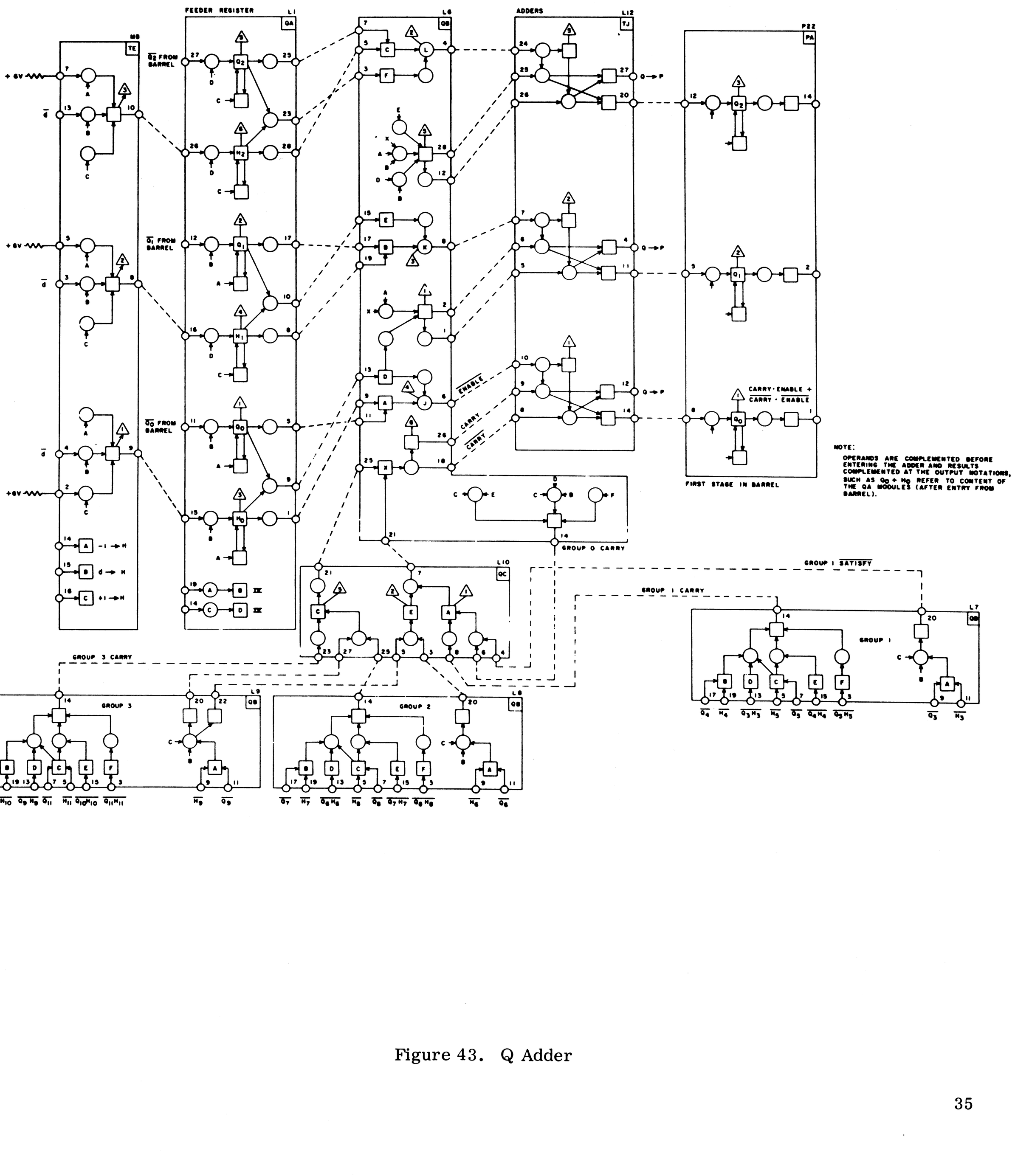
Subtractive adders are converse in the results when adding ± zero.

Example 6

+0	+0	-0	-0
++0	+-0	++0	+-0
+0	+0	+0	-0

Subtractive Adder Results

In a subtractive adder, obtaining negative zero as a result is only possible by adding negative zero to negative zero.



NOTE:

OPERANDS ARE COMPLEMENTED BEFORE ENTERING THE ADDER AND RESULTS COMPLEMENTED AT THE OUTPUT NOTATIONS, SUCH AS $Q_0 + H_0$ REFER TO CONTENT OF THE QA MODULES (AFTER ENTRY FROM BARREL).

Figure 43. Q Adder

Q Adder Operation

The A and Q adders in the peripheral processors operate in the same manner. In the following explanation the Q adder is used since the 12-bit word size is more convenient to work with than the 18-bit word size of the A adder.

Adder Operation

A discussion of general adder operations must be based on the type of adder being considered. Postulate a one's complement additive adder which makes use of carries. This adder is made up of the following parts: a set of circuits to determine the half add (partial add) of the two operands; a set of circuits to determine carries and carry propagation; and a set of circuits that combine the half add and the carries to determine the final answer.

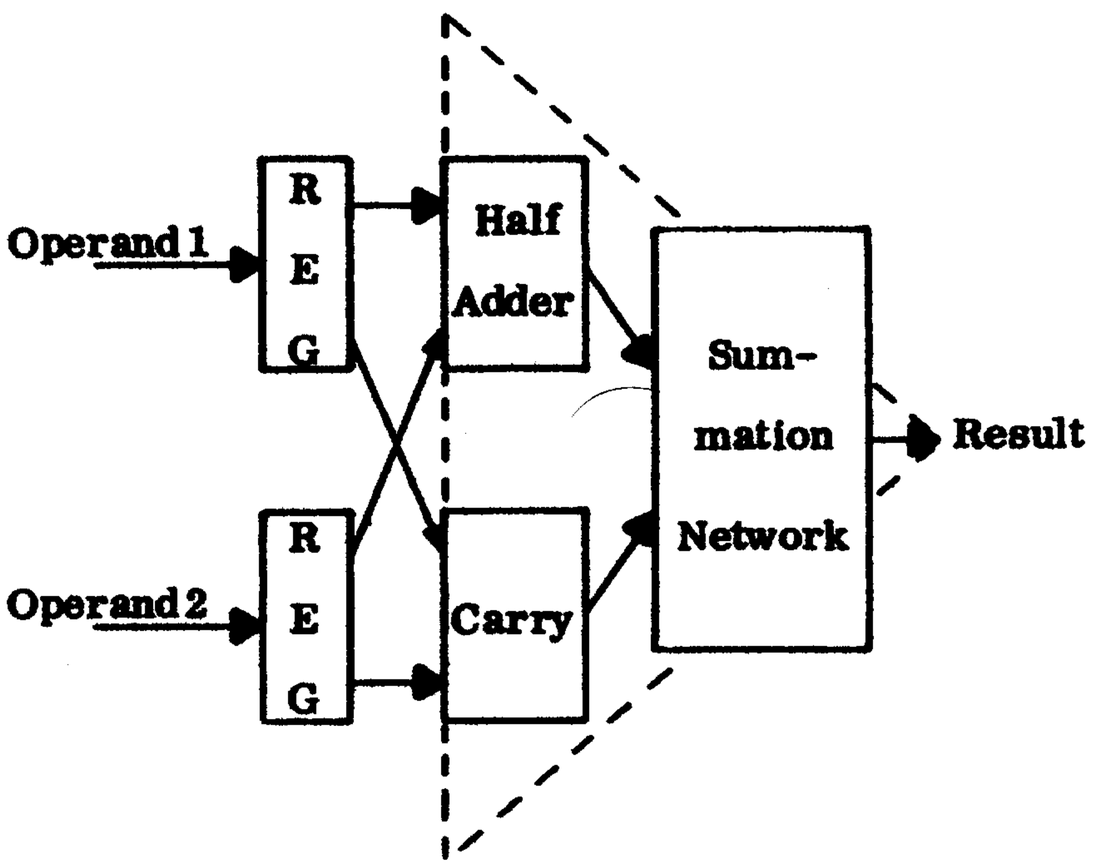


Figure 44. Adder Elements

Considering each component separately, we shall begin with the half-adder. The half-adder is concerned with only the partial add of the two operands and does not consider any carries; therefore, the half-adder must recognize the combinations shown in example 7.

Example 7

Operand 1	1	0	1	0
Operand 2	1	0	0	1
Partial Add	0	0	1	1

We can see that there are four possibilities for two operands but that there are only two possible results. Whenever the two operands are alike, whether both zeros or both ones, the result is a zero. Whenever the two operands are unlike, the result is a one. This can be simplified to having equivalence or not having equivalence to determine the half-add result.

Carry Network

Next, consider how carries must be handled in a 12-bit adder. Divide the adder into four groups of three bits each. Call each bit a stage and every three stages a group.

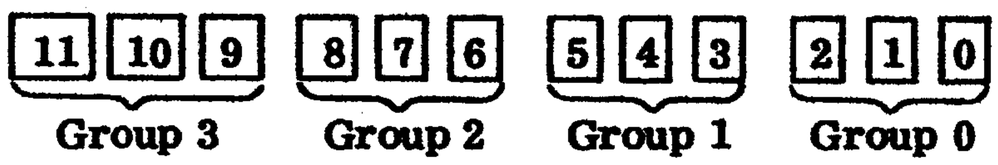


Figure 45. Carry Network

Determination of propagation and generation of carries is based on the same factors as are used in normal paper and pencil addition. That is, if you add seven and one octally, you obtain an answer of 10, a zero result for that digit position and a carry to the next position (example 8).

Example 8

Carry
XXX7
XXX1
0

Then we can say that in our adder, considering binary stages, that a one and a one is a generate, that a zero and a zero is a satisfy, and that a one and zero is a pass (enable).

Example 9

1	0	1 or 0
1	0	0 or 1
Generate (G)	Satisfy (S)	Pass (P)

One example of carry propagation we must consider is the end-around carry (EAC). This occurs when a carry from group three is sent to group zero.

This end-around carry could have been generated in any of the groups and not satisfied. Example 10 illustrates all possibilities.

Example 10

$$EAC = (GG0)(\overline{GS1})(\overline{GS2})(\overline{GS3}) + (GG1)(\overline{GS2})(\overline{GS3}) + (GG2)(\overline{GS3}) + (GG3)$$

Notice that the equation was written at the group level. What determines the condition of a group? For a group to be a generate, conditions shown in example 11 must be present.

Example 11

GG = SG2 X X + Case 1
SE2 SG1 X + Case 2
SE2 SE1 SG0 Case 3

} First Group Condition

The preceding equations state that a group generate condition exists if the highest stage is a generate, or if the highest stage is an enable and the second stage is a generate, or if the two top stages are enables and the lowest stage is a generate. There is a fourth condition where a group can be sending a carry upward but is not generating it. This occurs when a group is receiving a carry and is not satisfying it. The group is enabling, and it is called a group enable (example 12).

Example 12

(GE = SE3 SE2 SE1) Second Group Condition

The third condition is that a group could be satisfying. This occurs when the group satisfies a carry. A consideration must also be made when a group is both satisfying and generating carries. The group has a satisfy possibility, but it will generate a new carry at a different position.

In summary, the following rules can be made about carries and carry propagation:

- A carry will be generated by a one and one condition
- The carry will propagate upward until it is satisfied
- Carries must be considered within a group and also between groups

Final consideration is to the carry and half-add summation network. This network must make a decision, bit-for-bit, on the results of the stage equivalence and the possibility of a carry. The ensuing carry into a stage will complement the results of the half-add as shown in example 13.

Example 13

Operand 1	1	0	1	0
Operand 2	1	0	0	1
Results with No Carry Input	0	0	1	1
Results with Carry Input	1	1	0	0

The four equivalence/carry possibilities are equivalence with a carry, equivalence with no carry, no equivalence with a carry, or no equivalence with no carry (example 14).

Example 14

$(E)(\overline{C}) = 0$
$(E)(C) = 1$
$(\overline{E})(\overline{C}) = 1$
$(\overline{E})(C) = 0$

The following operands can be added in accordance with the preceding rules.

Example 15

Operand 1	1234			
Operand 2	5670			
	7124			
	ESG	EGS	EGG	ESS
Operand 1	001	010	011	100
Operand 2	101	110	111	000
Equivalence	011	011	011	011
Carry Inputs	011	101	110	000
	111	001	010	100

The two numbers can be added by first writing the two operands in binary notation. Then, by using the rule for equivalence, perform the half-add. At the top of each stage position, mark down the condition of that stage so that those which have carry inputs can be determined. Once all stages that have carry inputs are known, form the final answer by comparing the condition of the half-add with the status of the carry input.

Q ADDER

Now compare the Q adder to the one just presented. The first component of the Q adder decides on the equivalence or not equivalence of each particular stage of the operand.

Figure 46 illustrates the equivalence circuit. The inputs are the two operands to be added. As these inputs move through the circuit, equivalence is determined. The input to inverter X is $(Q+H)(\overline{(Q)(H)})$.

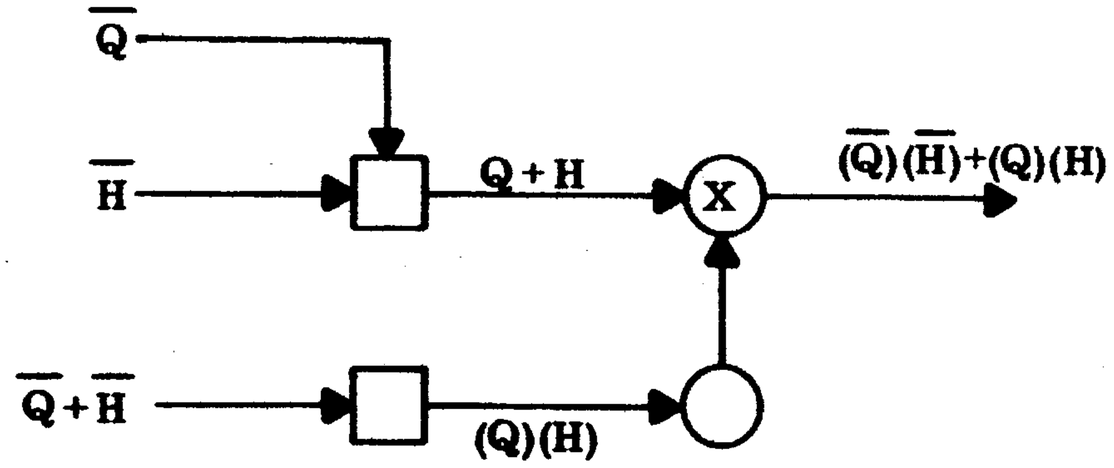


Figure 46. Equivalence Circuit

To determine the condition which produces a logical one output of inverter X, the two inputs are ANDed and then NOTed as follows: $\overline{(Q+H)\ ((Q)(H))}$. The answer can be found by working out the equation as shown in example 16.

Example 16

Inverter X Output = $\overline{(Q+H)\ \overline{(Q)(H)}}$

Simplification	Equation	
First	$\overline{(Q+H)}+\overline{\overline{(Q \cdot H)}}$	DeMorgan's Theorem
Second	$\overline{(Q+H)}+(Q \cdot H)$	Double Negation
Third	$(\overline{Q} \cdot \overline{H})+(Q \cdot H)$	DeMorgan's Theorem
Result	$\overline{Q} \cdot \overline{H} + Q \cdot H$	Equivalence

Now that the equivalence has been found, the next component to examine is the carry circuity. Since the adder is symmetrical, you can examine the circuits for one group, knowing that these same circuits are repeated. For example, examine the carry input circuit for group 0.

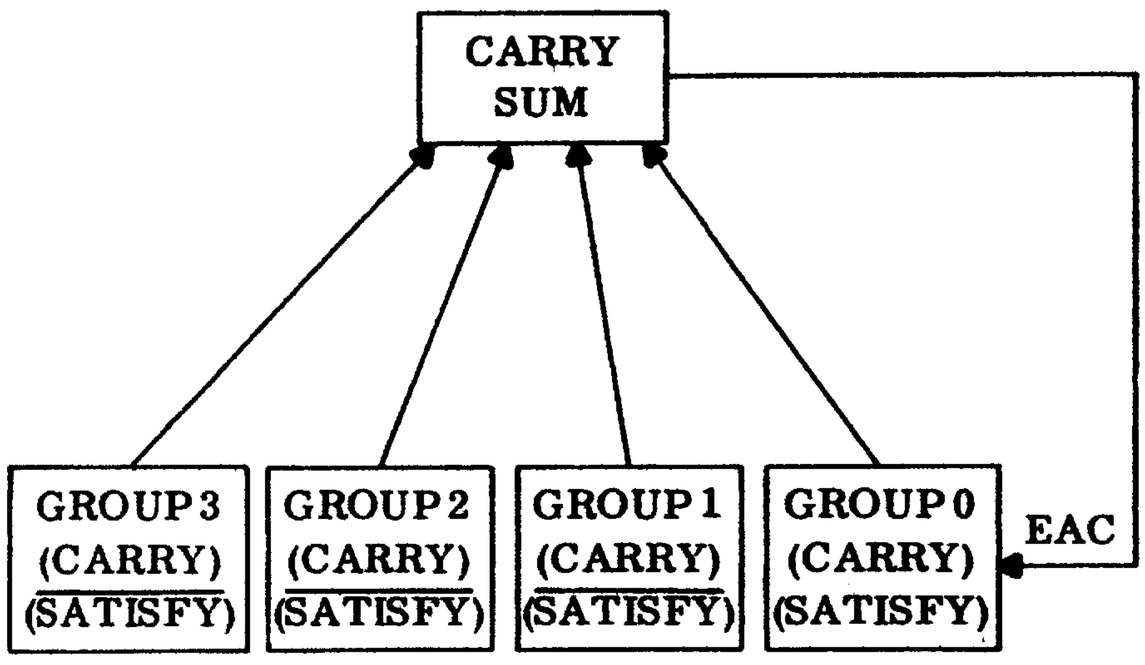


Figure 47. Carry Input Circuit for Group 0

The complete circuit consists of the group condition circuits for all four groups. These are located on L6, L7, L8, and L9. All of these circuits feed a carry summation network located on L10 which determines if a carry is to be sent to group 0. The above stated configuration appears four times for the Q adder.

Looking at the module that decides the group condition (such as module L7) we see inputs coming from all stages in that group. Example 17 lists the conditions in Boolean notation.

Example 17

Inverter A = Q3 + H3
Inverter B = Q4 + H4
Inverter C = Q5 + H5
Inverter D = (Q3)(H3)
Inverter E = (Q4)(H4)
Inverter F = (Q5)(H5)

Therefore, pin 14 = ((Q5)(H5)) + (Q5 + H5) ((Q4)(H4)) + (Q4 + H4) ((Q3)(H3)) (Q5 + H5). The above equation basically says that in order to get a one out from pin 14, any one of three conditions must exist.

- The most significant stage is a generate
  $((Q5)(H5)) = \begin{matrix}1\\1\end{matrix}$

- The most significant stage is an enable
  $(Q5+H5) = \begin{matrix}1 & 0\\0 & 1\end{matrix}$ and the second stage is a generate $((Q4)(H4)) = \begin{matrix}1\\1\end{matrix}$

- The most significant two stages are both enable
  $(Q5 + H5)(Q4 + H4) \begin{matrix}1 & 0\\0 & 1\end{matrix}$ and the lowest stage is a generate $((Q3)(H3)) \begin{matrix}1\\1\end{matrix}$

Therefore, if pin 14 is a one, group 1 is generating a carry. If this carry cannot be satisfied in a higher group, it will develop an EAC.

A second signal that must be generated from L7 indicates whether group 1 can satisfy an incoming carry. Pin 20 indicates the satisfy condition for group 1. Using Boolean notation, pin 20 = (Q5 + H5) (Q4 + H4) (Q3 + H3).

This equation states that there is at least one one in every stage. This indicates that there are no satisfies present in this group since a satisfy is a $\begin{matrix}0\\0\end{matrix}$.

The outputs of L7 (pins 20 and 14) go to the carry summation circuit on module L10, accompanied by similar signals from all other groups. The outputs of L10 (pins 21 and 7) determine if a carry enters stage 0. A zero at either pin indicates a stage 0 input carry (EAC). Example 18 shows the conditions after examination of the circuits on L10.

Example 18

Inverter C = GG3 Carry + GG2 Carry $\overline{(GG3\ Satisfy)}$
Inverter E = $\overline{(GG3\ Satisfy)}\overline{(GG2\ Satisfy)}$
Inverter A = GG1 Carry + GG0 Carry (GG1 Satisfy)

If pin 21 is a zero, indicating a carry, the situation is group 3 generating, or group 2 generating and group 3 not satisfying. If pin 7 is a zero, the situation is group 3 not satisfying, group 2 not satisfying, and group 1 generating; or group 0 generating and group 1 not satisfying.

Stage Carry

Up to this point, the discussion has concerned groups. Now we shall examine the effect of carries within the group between stages. Examining a typical

module, such as L6, we find three stages of the Q adder. The term X represents a carry input to the group. The first situation considered is what occurs if the lowest stage receives a carry input (X). This can only happen with an end-around carry condition. For the second stage to have a carry input, the lowest stage must generate a carry, or a carry must be coming in and the lowest stage is not satisfying.

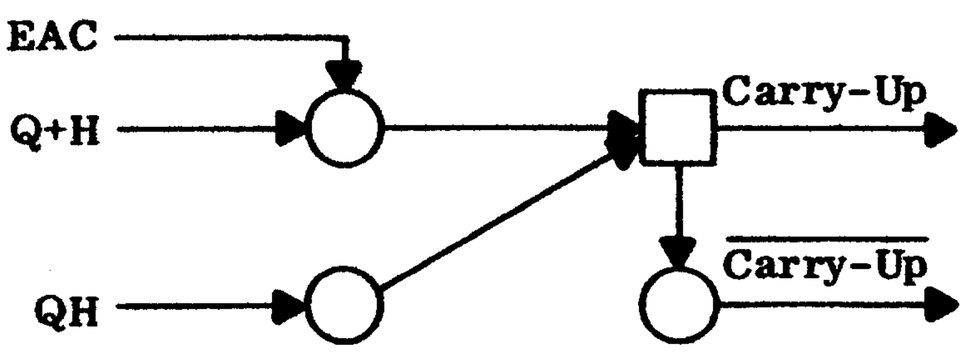


Figure 48. Carry-Up

Figure 48 shows that to get a carry output to the second stage, either Q and H must both be ones, indicating a generate, or an EAC must be present and stage 0 must not be a satisfy (Q+H). For a carry input to reach the third stage of a group, the following conditions must be present. First, the second stage is generating a carry (QH); second, the first stage is generating a carry (QH) and the second stage is not satisfying (Q+H); or, third, there is an EAC and the first and second stages are enabling (Q+H).

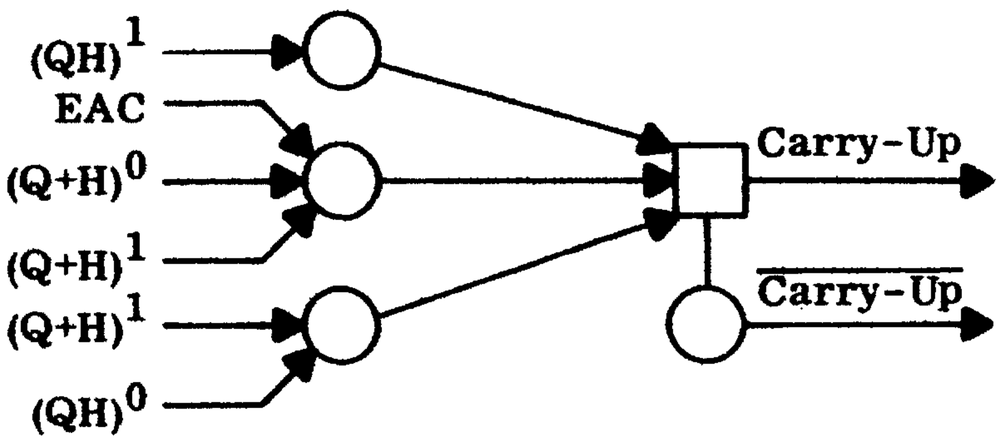


Figure 49. Carry-Up

Adder Summation Network

The last component of the Q adder to be considered is the adder summation network. This circuit must consider the results of the equivalence network and the carry input for each stage to determine the final answer.

Example 19

$(C)(\overline{E}) = 1$
$(E)(\overline{C}) = 1$
$(\overline{E})(\overline{C}) = 0$
$(E)(C) = 0$

The rules illustrated in example 19 are used after determining the equivalence and carries using the complemented operands that are in the feeder registers.

K REGISTER

K in the slot consists of a three-bit trip counter for the lower three bits (figure 52) and a fan-in for the upper six bits. The Advance K signal to the trip counter is enabled by instruction translations. For some instructions, the Advance K signal is controlled by signals which indicate status, i.e., the 5X0 trip is skipped by all 5X instructions if d=0. And, when K = 732, K is advanced only if the I/O channel is empty and active and A=1.

The three-bit trip count controls the sequence of operations for each instruction and is sometimes changed by gates other than the trip counter. For instance, for a Central Write instruction (63), K is changed from 637 to 633 to repeat the sequence of commands and send another word. When a 63 instruction is completed, K is changed from 637 to 733 to finalize the instruction and obtain the next instruction from storage.

The fan-in to the upper six bits of K allows the instruction code (F), to be entered into K from storage. The K⟶K path allows another trip around the barrel for the present instruction. The path K=340 is used by Replace instructions which automatically use the Store instruction (34) to accomplish the store portion of the Replace instructions.

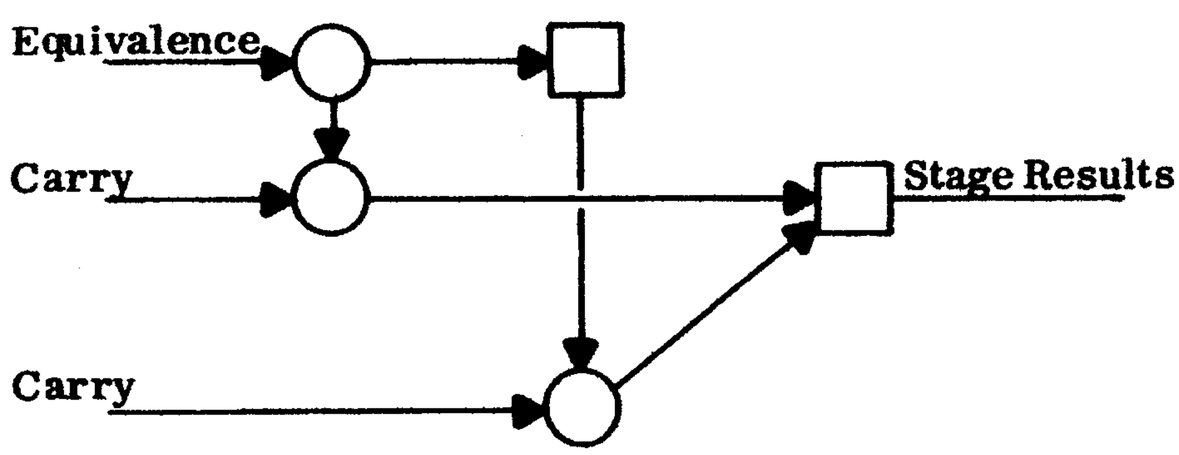


Figure 50. Summation Network

One point that must be brought out at this time is the definition of the Q adder. The Q adder is defined as a subtractive device. The only way of generating a negative zero in this device is to add negative zero to negative zero. However, all of our discussion has been based on an additive adder. The adder is actually additive but by inverting the input operands and inverting our final results, the adder appears as a subtractive device.

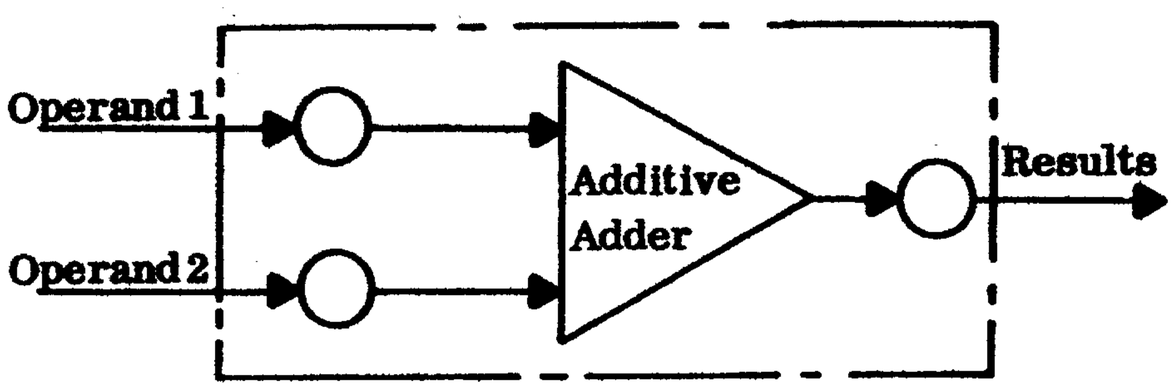


Figure 51. Q Adder

The inversion principle can be proved by using certain test operands and proceeding as the adder device actually operates (table 6).

We have shown that the results of step 4 in table 6 are the desired results because we inverted our input and the final answer before sending it to the barrel.

Considering the necessity for the preceding inversion, we must now alter our original equation for determining the final result of the adder. The rules are listed in table 6.

Table 6. Adder Progression of Two Test Operands

Step Number	Test Operands	Case 1	Case 2	Case 3	Case 4	Case 5
1	1	0000	0000	7777	7777	2525
	2	0000	7777	0000	7777	5252
2	Desired Results	0000	0000	0000	7777	7777
3	Additive Adder Results	0000	7777	7777	7777	
4	Inverter Input and Final Results					
	Inverter Input	7777	7777	0000	0000	
		7777	0000	7777	0000	
	Adder Result	7777	7777	7777	0000	
	Inverter Output Results	0000	0000	0000	7777	

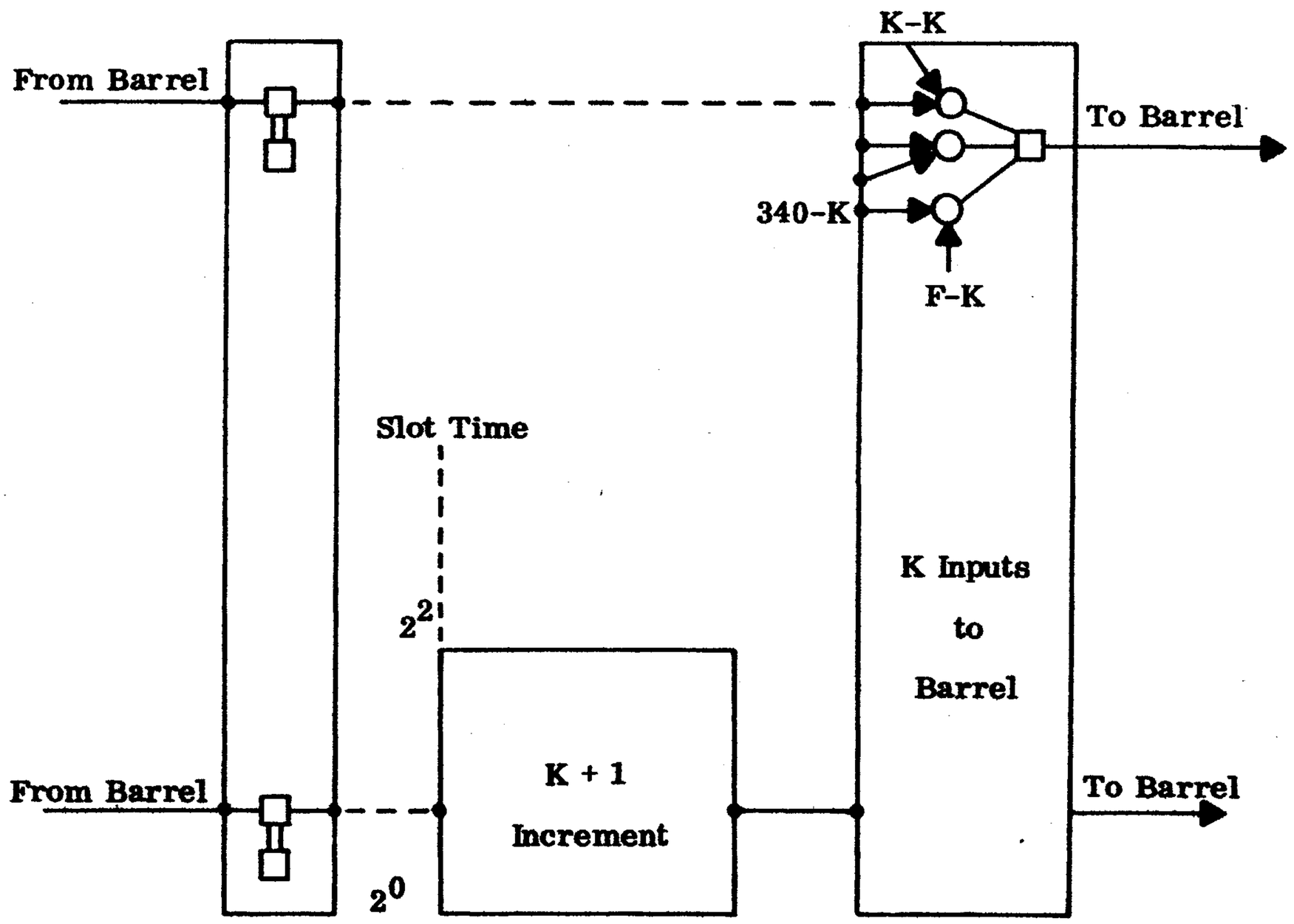


Figure 52. K in Slot

Each of the 12 independent I/O channels can handle 12-bit words at a maximum rate of one word every major cycle (1 mc rate). Each channel has an active/inactive FF and a full/empty FF (figure 53) which indicate channel status to the processors. Any channel can be used by a processor, but the external equipment assigned to a channel is wired in and can be assigned to another channel only by changing cable connections.

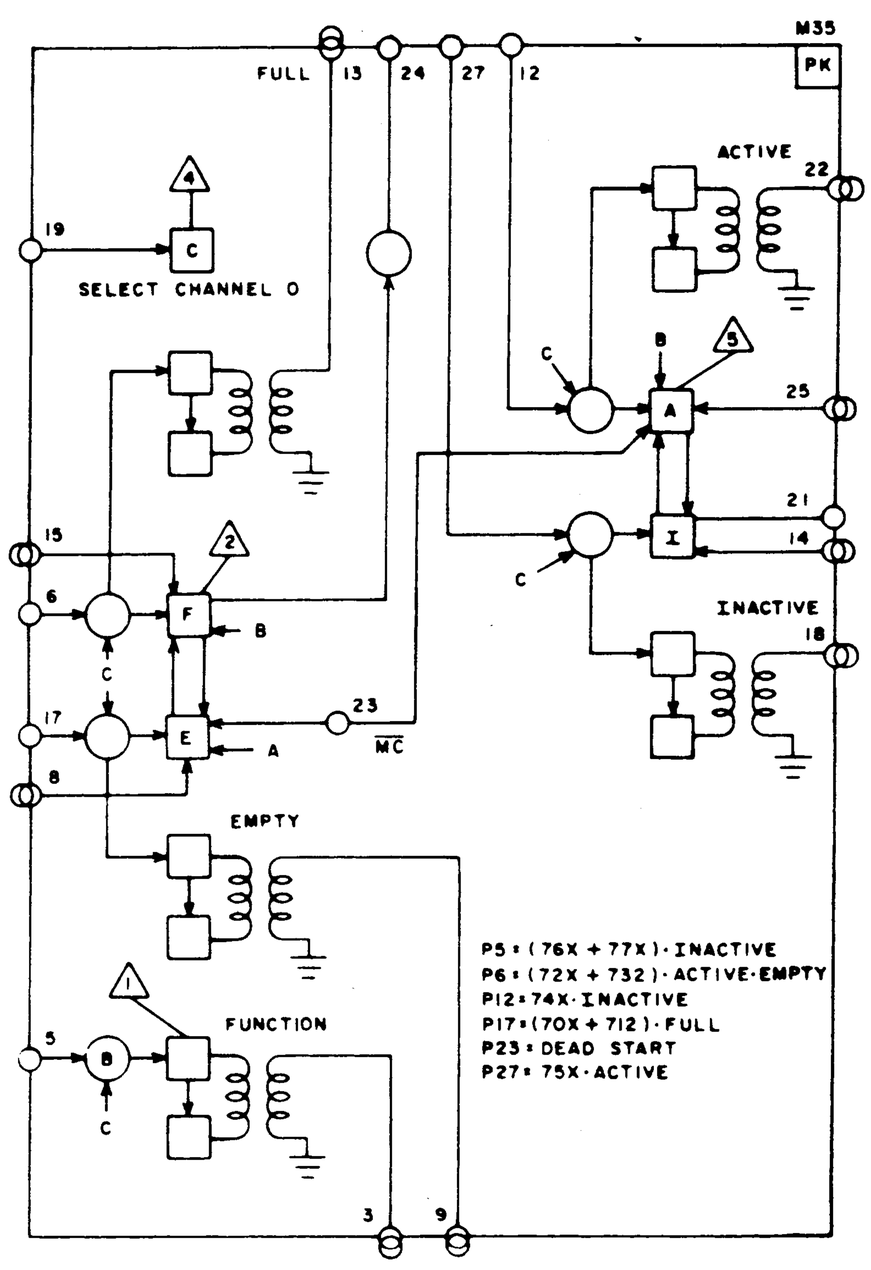


Figure 53. Channel Controls

The lines of an I/O channel are shown in table 7.

Table 7. I/O Channel Lines

Input	Output
Data or Status Reply (12 bits)	Data or Function Word (12 bits)
Active	Active
Inactive (Disconnect)	Inactive
Full	Full
Empty	Empty
	Function

Two clock signals are available to external equipment. They are a 1 mc clock and a 10 mc clock. The clock pulses are 25 ns wide, as are all data and control signals (except Master Clear). A synchronizer for each external equipment (or group) changes the I/O pulse signals into the signals required by I/O devices and changes input signals to the pulses required by the I/O channels. Synchronizers are described in a separate manual.

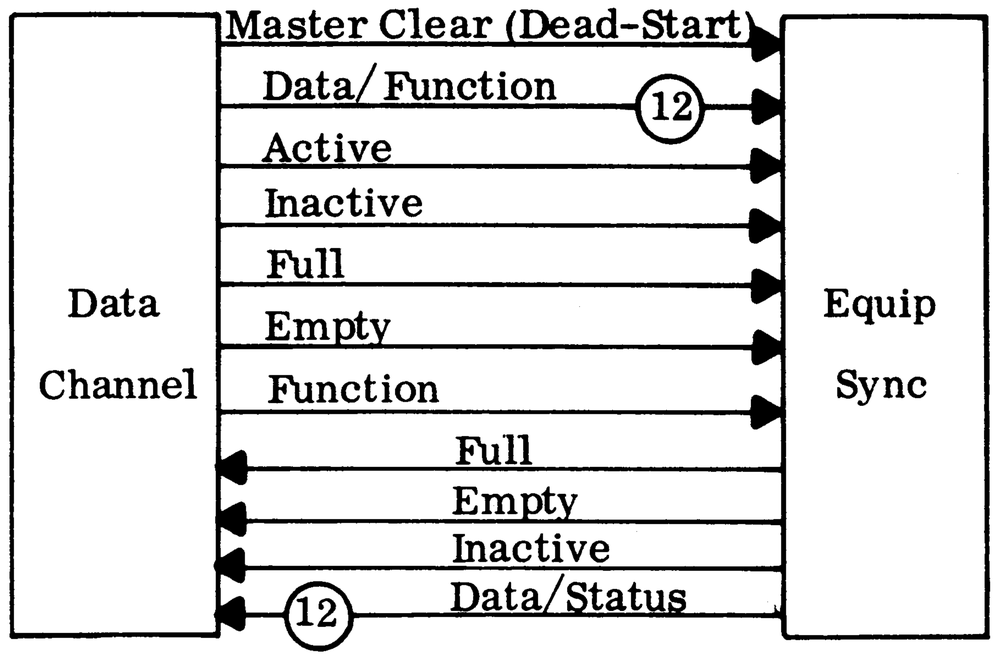


Figure 54. Channel Lines

An I/O channel can be used for communication between processors if it is selected for input by one processor and for output by another. The status of I/O channels can be sensed by instructions 64 - 67: Jump to m if Channel D Active, etc.

I/O INSTRUCTIONS AND SIGNALS

MASTER CLEAR

A Master Clear (MC) signal is generated only by the dead start circuit. MC removes all equipment selections (except dead start synchronizer) and sets all channels to the active and empty condition (ready for input). MC is a one μsec pulse which is repeated every 4096 ms while the Dead Start switch is on.

DISCONNECT (75)

A Disconnect instruction clears the channel active FF, if it is set, and sends an inactive pulse to the equipment on that channel. If a Disconnect instruction is given for a channel which is already inactive, the processor which issued the disconnect hangs up and is not able to continue to another instruction until the channel is activated by another processor or by MC.

FUNCTION (76 OR 77)

A Function instruction sends a 12-bit function code (from A or Fd) on the data lines and also sends a Function signal. It also sets the active and full FFs

for the channel but does not send active and full pulses. During a function, Active and Full are important to the channel but not to the external equipment. Upon receipt of the function code, the external equipment sends an Inactive (Disconnect) signal which clears the active FF which, in turn, clears the full FF. If a function instruction is given for an active channel, the processor hangs up until the channel is deactivated.

ACTIVATE (74)

An Activate instruction sends an Active signal on the channel and sets the active FF if the channel is inactive. If an Activate instruction is given for a channel which is already active, the processor which issued the instruction hangs up until the channel is inactivated by another processor or by an Inactive (Disconnect) signal from an external equipment on the channel.

INPUT (70, 71)

An Input instruction (70, 71) can call for an input of one word to A (70) or a block of words to memory (71). A block can be as small as one word. For either instruction, the control signals sent to and received from the external equipment are similar. A 70X or 712 translation clears the full/empty FF and sends an

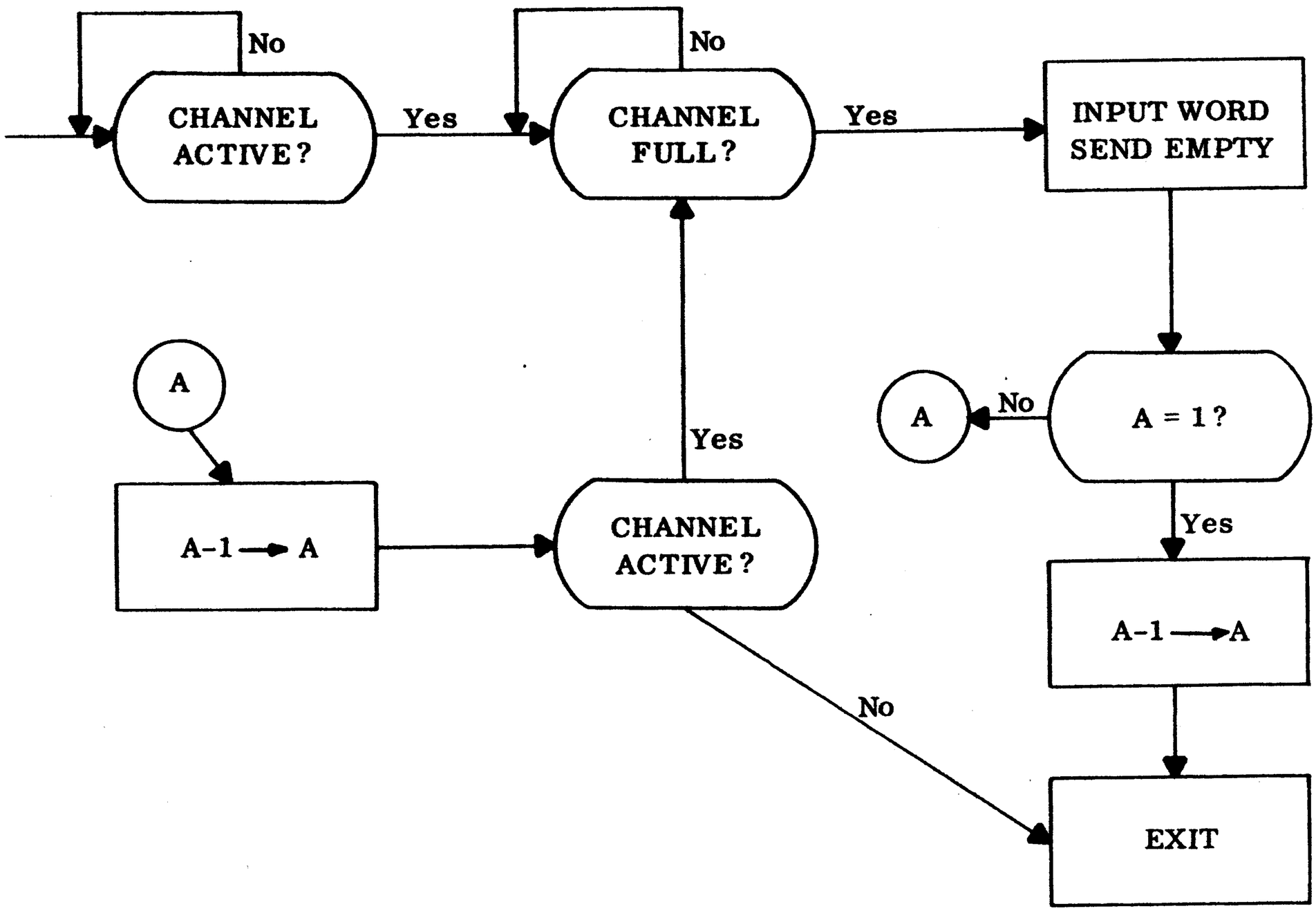


Figure 55. Input Operation

Empty signal on the designated channel. When the input device sends a word on the channel, it also sends a Full signal which sets the full/empty FF.

Full and Active signals enable the processor to store the input word or send it to A. A 70 instruction sends the input word to A and exits to the next instruction. A 71 instruction stores the word in memory, reduces the count in A by one, and sends an Empty signal to the channel. A 71 instruction continues to read words until the word count in A is reduced to zero. The processor sends one more word when it senses that the previous read operation reduced (A) to 1.

OUTPUT (72, 73)

Output instructions (72 and 73) transmit one word from A or a block of words from memory on an I/O channel. A block can be as small as one word.

An instruction translation of 72X or 732 and Empty and Active sets the full/empty FF and sends a full pulse and a data word on the channel. When the external equipment accepts the word, it sends an empty pulse which clears the full/empty FF. Empty and Active enable the processor to continue the instruction. A 72 instruction does not wait for the Empty signal but exits to the next instruction after sending one word.

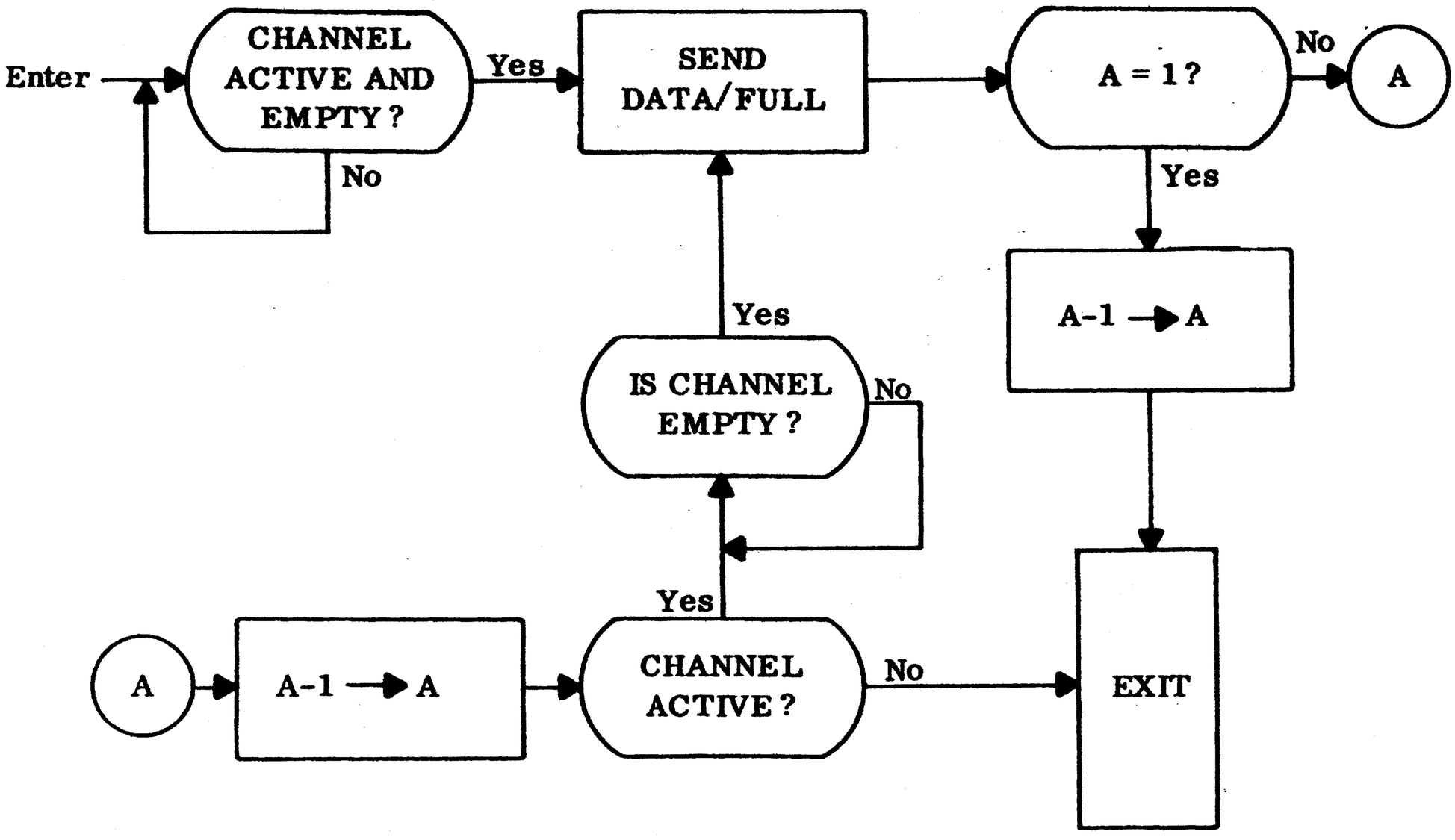


Figure 56. Output Operation

When the Empty signal is returned on a channel which is being used for a 73 instruction, the processor reduces the word count in A by one and sends another word. A 73 continues to send words accompanied by full pulses until the required number has been sent. When the processor senses that A has been reduced to one, it sends one more word, obtains the address of the next instruction, and continues with its program.

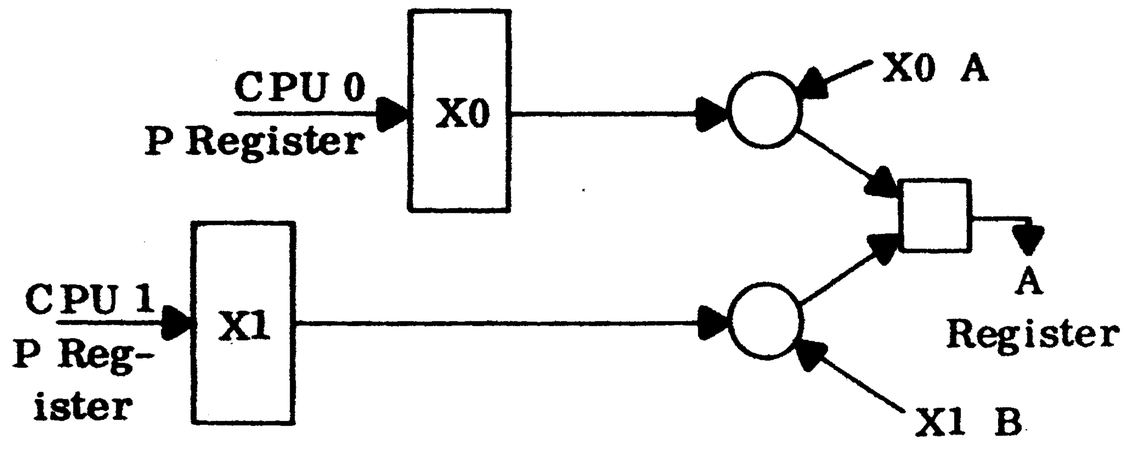


Figure 57. Central P Address to A

COMMUNICATION WITH CENTRAL MEMORY AND CENTRAL PROCESSOR

The peripheral and control processors can communicate with the central processor and central memory in several ways. They can read the central processor's program address, tell the central processor to jump to a given central memory address for its next instruction, or read from or write into central memory.

6400 ONLY CENTRAL PROGRAM MONITOR

An 18-bit central processor program address is sent from each processor to the central program monitor register X0 or X1 (figure 57) on chassis 1 every minor cycle. A Read Program Address instruction (2701 + 2700) sends the central address to the A register. Thus, the progress of a central program can be monitored by any peripheral and control processor.

Exchange Jump, Central Read and Central Write instructions all use (A) as a central memory address. (A) is unconditionally sent to address control in the central processor every minor cycle (figure 58). This address is recognized and used as a central memory address only if accompanied by a Central Read, Central Write, or Exchange Jump signal.

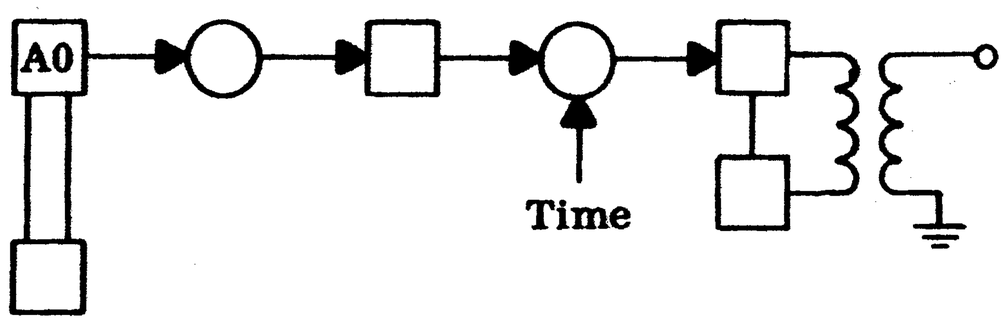


Figure 58. (A) to Central Memory

CENTRAL BUSY

A central busy FF (figure 59) indicates when a reference to central is in progress. A central busy condition prevents initiating a central reference until the one in progress is completed.

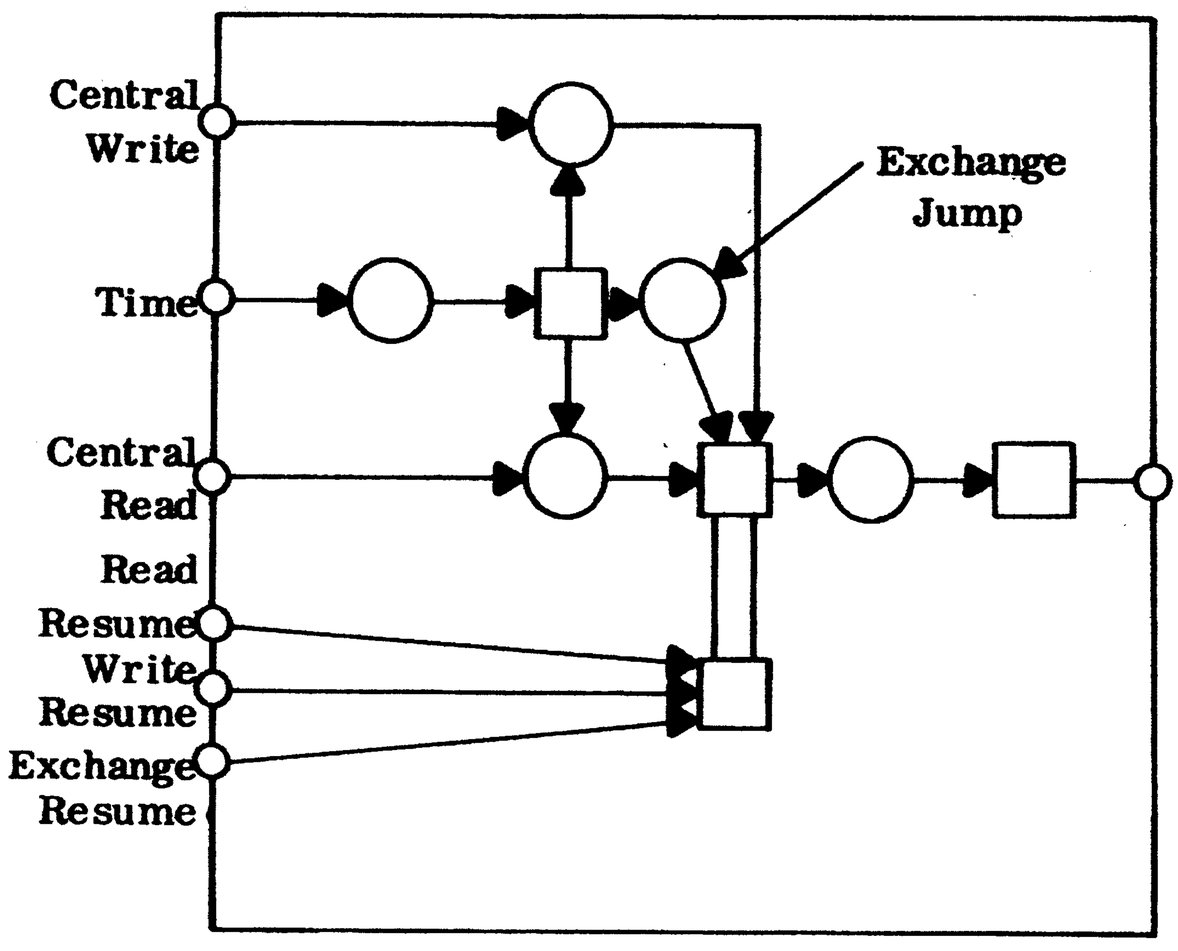


Figure 59. Central Busy

EXCHANGE JUMP

An Exchange Jump instruction is used to command the central processor to stop the program it is executing and go to a central memory location specified by the instruction. An Exchange Jump can be issued by any peripheral and control processor as long as the central busy FF is clear. The instruction sends an Exchange Jump signal to the central processor and sets the central busy FF. The Exchange Jump signal tells the central processor to recognize the 18-bit address sent from the peripheral processor and to perform an Exchange Jump. After the central processor has performed the Exchange Jump and started a new program, it sends a Resume signal which clears the central busy FF. This must be done before another central reference can occur. If a peripheral and control processor tries to issue an Exchange Jump instruction while the central busy FF is set, the processor must wait until the previous central reference is completed and the central busy FF is cleared.

CENTRAL READ

The Central Read instruction allows a peripheral and control processor to obtain one word (60-bits), or a block of words, from central memory. The instruction sends a Central Read signal to central address control enabling it to use the 18-bit quantity from A as a central memory address. At the same time, the central busy FF is set to inhibit other references to central until the read word is received. When a 60-bit word is sent by central to the central read pyramid (figure 60), it is accompanied by two control signals, an Accept signal which clears the central busy FF, and a signal which sets the C5 full FF. Each rank of the central read pyramid (C1 through C5) has an associated full/empty FF used to control the flow of data through the pyramid. C5 full and C4 empty enable the processor which is doing the Read instruction to send the upper 12 bits of C5 to memory and the lower 48 bits to C4. Subsequent steps in the Central Read instruction step the central word down through the pyramid and store the rest of the central word as 12-bit peripheral words. Each step in this storage procedure requires that the next lower rank in the pyramid be empty before a transfer is made. No Central Read instruction can be issued until C5 full and central busy FFs are clear. However, as many as four central memory words, in different stages of disassembly, can be in the central read pyramid at one time. A Read instruction, for which the proper full and empty conditions are not met is not processed until previous instructions progress further and the necessary conditions are met.

A 60 instruction reads only one central memory word and stores it as five peripheral words. A 61 instruction reads a block of words specified by (d). In either instruction, the first central memory address is specified by (A). For a 60 instruction, d specifies the peripheral address at which the upper 12 bits of the peripheral word are stored; the next lower 12 bits go to d+1, etc. For a 61 instruction (d) gives the number of central words to be read, and m is the address for the upper 12 bits of the first central word.

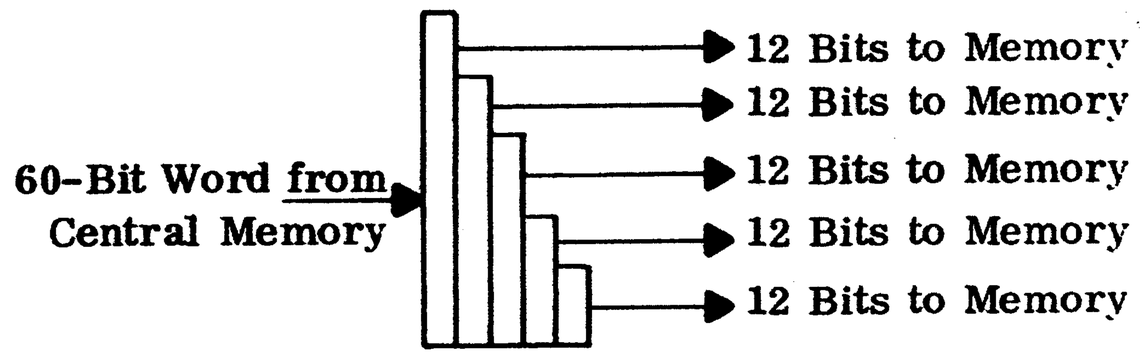


Figure 60. Read Pyramid

CENTRAL WRITE

Central Write instructions send one 60-bit word or a block of 60-bit words to central memory. Each 60-bit word sent to central memory is assembled in the central write pyramid (figure 61) from five 12-bit peripheral words. A Central Write instruction assembles a 60-bit word and sends the word and a Central Write signal to central address control and sets the central busy FF. The Central Write signal enables central address control to accept the 60-bit word and

store it at the address specified by (A). When the word has been stored, an Accept signal is sent back to clear the central busy FF. Up to four Central Write instructions may be in progress at one time with portions of four different words in D1 through D4. D5 is an output network only and cannot store a word. The first 12-bit word goes to D1 and is the upper 12 bits of the 60-bit word. When a second 12-bit word goes to D2, D1 is also sent to D2. When the fifth word goes to D5, the 48 bits in D4 are also sent to D5 and the 60-bit word is sent to central.

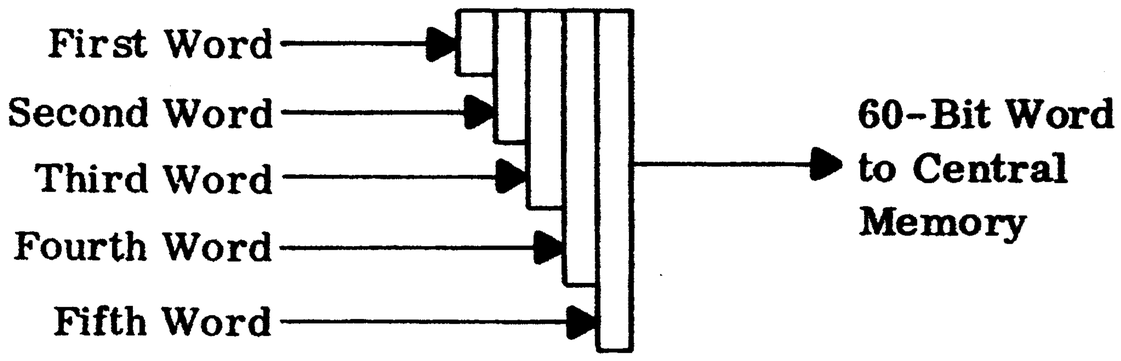


Figure 61. Write Pyramid

DEAD START

Dead Start is a system used to initially start the computer, dump the contents of the peripheral and control processor memories to a printer or other output device, or sweep memory without executing instructions.

The dead start panel contains a 12 x 12 matrix of toggle switches, a Sweep-Load-Dump switch and a Dead Start switch. It also contains memory margin switches which are used for maintenance checks.

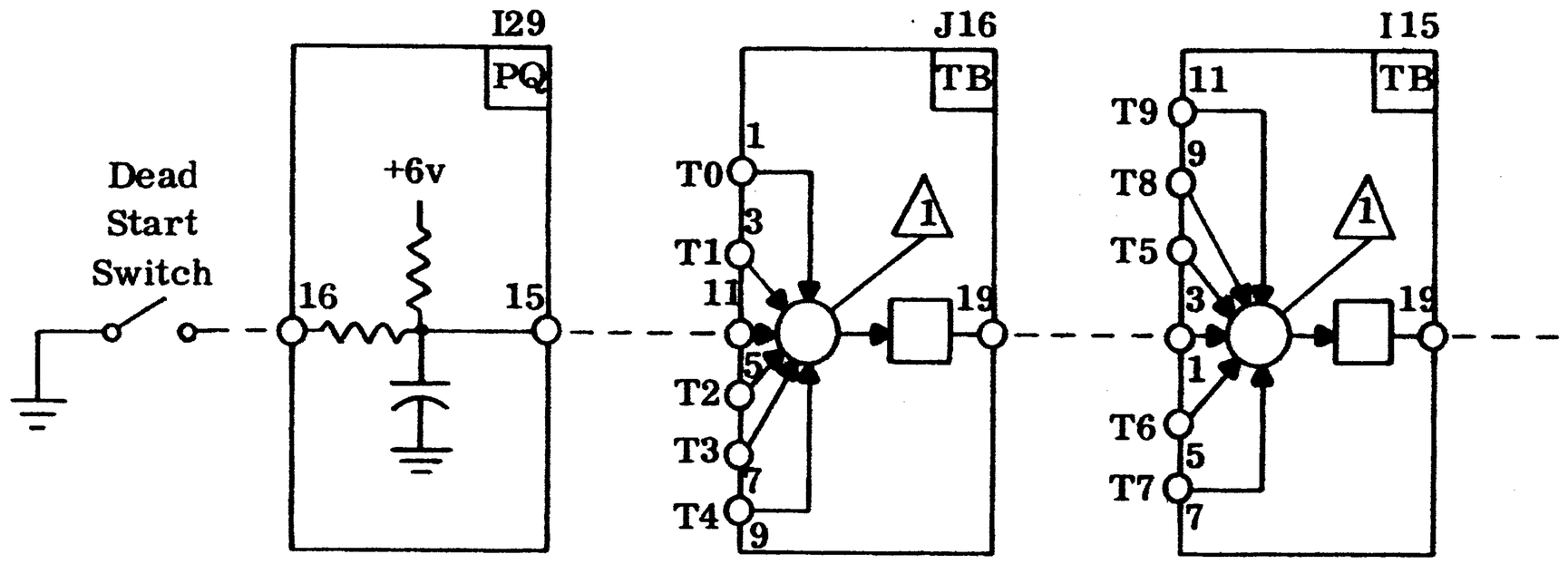
LOAD

To initially load programs and data into the 6400, the Sweep-Load-Dump switch is set to Load. The matrix of toggle switches is set to a 12-word program (up position equals one, down position equals zero). When the Dead Start switch is turned on a 1 μsec dead start pulse initiates the following actions.

- Assigns to each peripheral and control processor the corresponding I/O channel
- Sets all I/O channels to active and empty
- Sets K for all processors to 712 (input)
- Sends a MC on all I/O channels
- Sets P for all processors to zero
- Sets A for all processors to a word count of 10,000

The dead start pulse is repeated every 4096 μsec while the Dead Start (DS) switch is on. To start the 6400, the DS switch is normally turned on momentarily, then turned off. Recycling of the dead start pulse is controlled by the Real-Time clock. The pulse is formed by ANDing DS switch On with the 12 bits of the Real-Time clock.

When the dead start synchronizer on channel 0 receives the MC sent by dead start, it sends a full pulse but no data. When processor 0 receives the full, it stores the content of the channel 0 Input register (all zeros) in location 0000 and sends an empty pulse to the dead start synchronizer. The dead start synchronizer then acts like an input device and sends 12 words



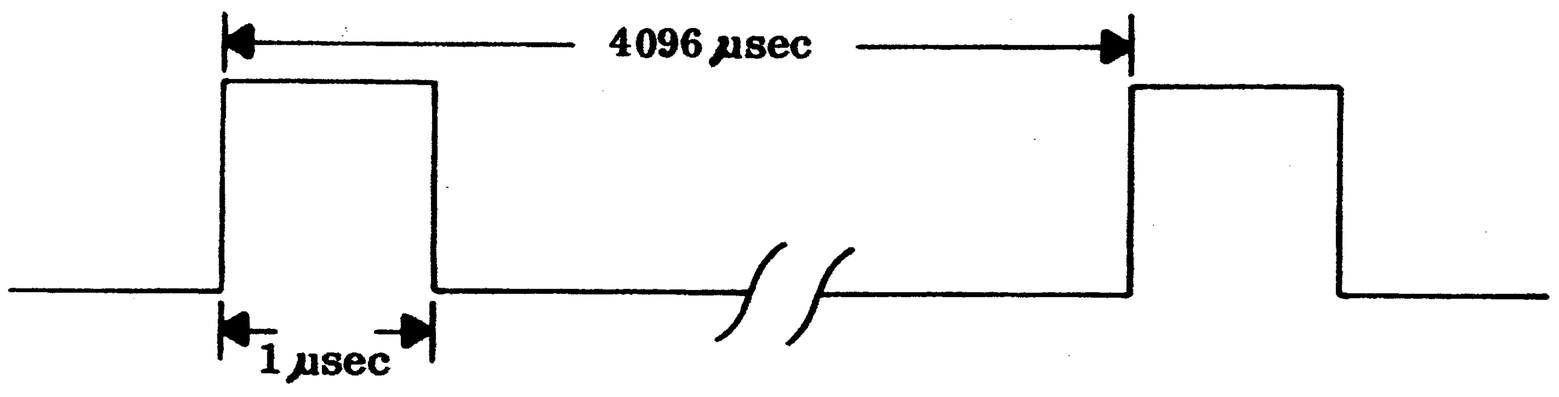


Figure 62. Detailed Timing for Dead Start Pulse

from the switch matrix (as 12-bit words), and processor 0 stores them in locations 0001 through 0014_8. After the last word, the dead start synchronizer sends a Disconnect signal which causes processor 0 to exit from the 712 instruction and store zeros in address 0015. Processor 0 reads location 0000, adds one to its content and goes to 0001 for its next instruction. It then executes the 12-word (or less) program, which normally is a control program, which is used to load information and begin operation. The other processors are still set to 712 and may receive data from processor 0 via their assigned I/O channels using interchannel communications.

SWEEP

If the DS switch is operated with the Sweep-Load-Dump switch set to Sweep, all processors are set to a 505 instruction and P registers are set to 0000. Since the 50 instruction does not require five trips around the barrel, there is no logic to clear or advance K from 505. The 50X translation of K causes all processors to sweep through their memories, reading and restoring without executing instructions. This is a maintenance routine and can be used to check the operation of memory logic.

DUMP

The dead start pulse, with the Sweep-Load-Dump switch set to Dump, initiates the following actions.

- Sets all processors to 732
- Sends MC on all channels except channel 0
- Holds channel 0 active and empty
- Assigns each processor to its corresponding I/O channel
- Sets all A and P registers to zero

All processors sense the empty and active conditions of their assigned channels, output the content of their address 0000, set their I/O channels to full, and wait for an empty condition. All processors advance P by one and reduce A by one ($A=7776_8$). Channel 0, which is assigned to processor 0, is held to empty by the Dump switch. Therefore, processor 0 cycles through the 732 instruction until A=1 and then goes to memory location 0001 for its next instruction. Thus processor 0 can send its entire memory contents on channel 0, although no I/O device has been selected to receive it. Processor 0 is then free to execute a dump program which must have been previously stored in memory 0 (beginning at location 0001).

6400 PERIPHERAL PROCESSORS

The 6400 Peripheral Processors are almost identical with the 6600 PPU's previously described. The slight differences in the 26XX and 27XX instructions, the X registers, and a few module-type changes are explained in the following paragraphs.

26XX INSTRUCTION

The 260X instruction differs from its 6600 computer counter-part in that the lowest bit is used to determine which CPU to Exchange Jump. The 2^0 bit equaling zero means CPU 0 (chassis 2); the 2^0 bit equaling a one means CPU 1, (chassis 8) the special central processor.

The 261X instruction has the 2^0 bit CPU definition, and it is conditioned upon the contents of a flag FF in central memory control. The instruction may or may not Exchange Jump CPU X. The 261X is normally used only with an ECS System (see 6400/6600 ECS Training Manual). For further details see the section covering Exchange Jumps.

27XX INSTRUCTIONS

The 270X instruction differs from its 6600 computer counter-part in that the 2^0 bit selects the CPU. The 2^0 bit equaling zero means read P of CPU 0. The 2^0 bit equaling one means read P of CPU 1. CPU 0 is the standard CPU located on chassis 2; CPU 1 is the special purpose CPU located on chassis 8.

ADDITIONAL X REGISTER

The possibility of two central processor units necessitated two X registers, one for CPU 0 (X0, the original register) and one for CPU 1 (X1, the additional register).

MODULE CHANGES

Although a few slots were available, the addition of the X register, the selection circuit to choose either X register, the 2^0 and 2^3 bit translation, etc., necessitated changes in module types for more compactness.

The D5 transmitter modules were changed from PL's (six transmitters) to JQ's (ten transmitters). This allowed the ten modules to be reduced to six, a gain of four slots.

The A register transmitters were changed from PL's to JQ's as well, providing a gain of one slot and a few extra transmitter circuits for other uses.

The X1 register was added (three PI's) along with the three TE selection circuits. The master clock was removed from this chassis and put onto chassis 8. These changes and the slots that were previously open allowed the 6600 peripheral processors to be converted to 6400 PPU's.

CHAPTER 4

CONCEPT OF THE 6600 CENTRAL PROCESSOR

INTRODUCTION

The CONTROL DATA 6600 Series Computer System, through use of high-speed transistor logic and a design philosophy based on concurrent (or parallel) processing, is today recognized as the world's fastest and most powerful computer. The rapid throughput achieved by the 6600 system can be attributed in part to the concurrency that exists in several areas of the central processor.

MEMORY BANK PHASING

The central memory is divided into memory banks, each of which contains 4096_{10} 60-bit central processor words. A 131K central memory is composed of 32 such banks; a 65K memory has 16 banks. Since each bank has its own circuitry for the X and Y drive lines, inhibit lines, sense lines, and memory cycle timing, each is capable of operating independently. This, in turn, permits memory cycles to be phased (overlapped) by 100ns to effectively reduce minimum access time to 100ns. That is, a memory cycle is one μsec in duration, but 10 may be initiated each μsec as long as they are to different banks. The bank phasing scheme, in addition to the memory cycle which is in itself extremely fast, eliminates a great portion of the memory waiting time that is inherent in the majority of computers.

INSTRUCTION STACK

A group of flip-flop registers, referred to as the Instruction Stack, is provided in the 6600 for the purpose of holding an iterative sequence of instructions (a program loop). The Stack can hold a loop containing

up to 27 instructions (up to four instructions per word) which may then be executed without the need for instruction word memory references (RNI's).* Initially, the eight stack registers (I registers) are filled by reading instruction words from central memory. As each word is read into and executed from I^0 (figure 63), the preceding words move up in the Stack and a new word is entered into the first I register. When the Stack is filled, the movement of instruction words causes the top word (in 17) to be discarded. When instructions are being executed in the Stack (looping), no movement occurs and the stack information remains static. In this manner, the necessity of fetching each instruction from memory is eliminated during short loops. The memory-access time savings should be obvious.

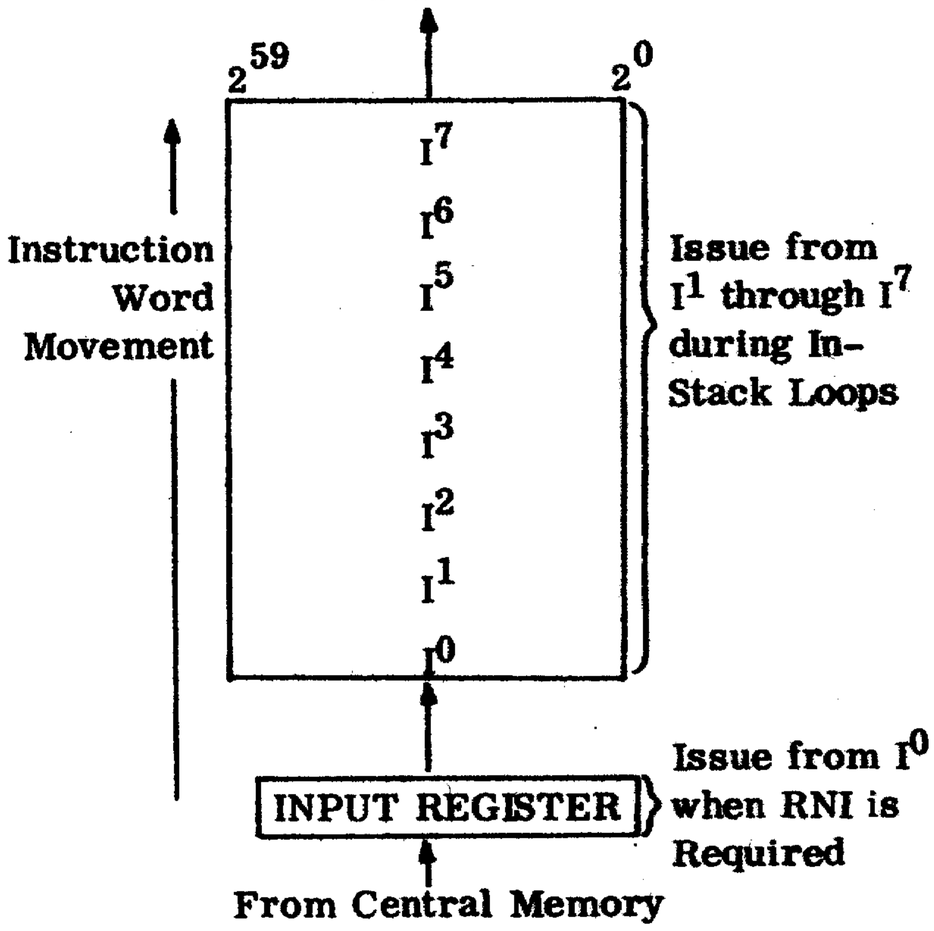


Figure 63. Instruction Stack

Operating Registers

Another property of the central processor decreases the number of memory references required to read and store operands. Twenty-four operating registers provide a flip-flop storage facility for 60-bit operands and 18-bit addresses and indexing values. Eight 60-bit registers (designated X0 through X7) provide for the storage of integer and floating point values in a 60-bit format. Eight 18-bit registers (designated A0 through A7) provide storage for central memory addresses of operands which are read or stored in memory. Eight 18-bit registers (designated B0 through B7) provide for storage of indexing values, used for modification of addresses and operands. Since most central processor instructions can designate two source operands (taken from X, B, or A registers), and one result designation (X, B, or A register), considerable operand manipulation can take place by use of the operating registers, thereby further decreasing the number of memory accesses needed. Proper use of the Instruction Stack and operating registers makes possible execution of program loops, which require no memory references, for instructions and operands or storage of results.

Functional Units

Another area of concurrency in the 6600 CPU is that of parallel arithmetic (functional) units. Ten logically independent functional units are provided to allow several instructions to be in various stages of execution at the same time. Table 8 describes the functional units and their corresponding cycle times.

Table 8. Cycle Times for Functional Units

Number	Unit	Time (ns)
1	Add (floating)	400
2	Multiply 1 (floating)	1000
3	Multiply 2 (floating)	1000
4	Divide (floating)	2900
5	Boolean (logical)	300
6	Long Add (integer)	300
7	Shift	300 - 400
8	Increment 1 (indexing)	300
9	Increment 2 (indexing)	300
10	Branch (branch instructions)	800 - 1400

Each unit is assigned a group of instructions which it, and only it, processes. For example, the Add unit processes all single precision, double-precision, rounded and unrounded floating point Add opcodes. The shift unit handles opcodes that require shifting: left and right shifts, normalize operations, packing, unpacking, etc.

Separate functional units eliminate the necessity for sequential execution of program steps, a property which is inherent in most present-day computers. Instead, unrelated instructions can be processed out of sequence, causing a considerable decrease in the overall execution time of a program. Of course, if a source operand for one unit is the result operand of another, the first unit must wait until the second

*Although 27 instructions may at first appear to limit the programmer's capability, it should be remembered that the 6600 is designed primarily as a scientific machine. Many programs are of a mathematical nature (i.e., matrix analysis). Also, each instruction can designate two source operands and one result destination. When viewed in this light, 27 instructions are, in most cases, more than adequate.

completes its calculation and returns the result. Also, if two division steps are needed in sequence, the second must wait until the first completes, since only one Divide unit exists. On the other hand, two multiply operations may take place concurrently because there are two Multiply units. The point to be stressed is that in most operational programs the instructions need not be executed in sequence. Instead, the majority of problems are composed of a series of smaller steps which are only indirectly related. The following programming comparison should illustrate this point.

The problem that follows is solved first by using a sequential computer and secondly, by using the 6600 with its functional units. Individual instruction execution times are assumed to be the same in both machines. Also, both have the capability of reading two source operands and returning one result using the operating registers (X, B, and A). Example 20 shows operating register contents. The ( ) means contents of.

Example 20

(X1) = the value, A

(X2) = the value, B

(X3) = the value, C

Initial Problem

$$\left(\frac{A + B}{C}\right) \bullet (A^2 + B^2 + C)$$

Resultant Problem

$$\left[\frac{(X1) + (X2)}{(X3)}\right] \bullet \left[(X1)^2 + (X2)^2 + (X3)\right]$$

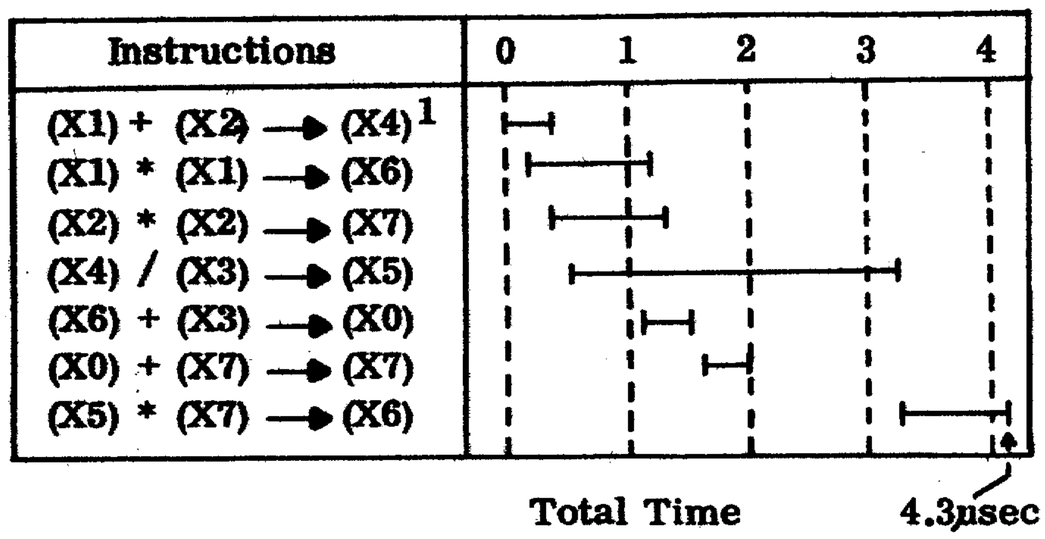
Table 9. Sequential Computer Instruction Execution Time

Instructions	Time (ns)
(X1) + (X2) → (X4)[1]	400
(X4) / (X3) → (X5)	2900
(X1) * (X1) → (X6)	1000
(X2) * (X2) → (X7)	1000
(X6) + (X3) → (X0)	400
(X0) + (X7) → (X7)	400
(X5) * (X7) → (X6)	1000
Total Time	7100

Since the instructions must be executed in sequence, the total execution time is the sum of the individual execution times, or 7.1μsec.

1 "→" means replaces

Table 10. 6600 Computer Instruction Execution Time



Using parallel functional units, the program execution time is only 4.3μsec, a reduction of approximately 40 percent.

Although the same saving will not occur in all programs, the example illustrates that, through efficient programming, a considerable decrease in execution time occurs. Even when a program is not optimized, some time saving will be realized.

SUMMARY

Several unique features are incorporated in the design of the 6600 CPU, including: 1) 32 (or 16) 4K, phased memory banks, 2) an Instruction Stack containing 8 60-bit registers, 3) 24 operating registers, and 4) 10 independent functional units. These provisions work in conjunction with each other to provide extremely rapid program execution times. Whenever parallel processing capabilities are provided in a computer, control circuitry is required to ensure that all features work together (without calamity) to produce a high-speed processing system.

BLOCK DIAGRAM ANALYSIS

CENTRAL MEMORY ADDRESS CONTROL

References to central memory can be initiated from various sources in the 6600. Peripheral processors make central memory references during the Central Read, Write, and Exchange Jump instructions. The central processor uses central memory to fetch instruction words or to read and store operands. An orderly means for handling these memory requests and distributing the associated data must be utilized. This is complicated by the fact that the 6600 memory banks are phased to allow several memory cycles to be in progress at any one time. Therefore, it is very possible that a memory reference request could be made to a bank that is already busy processing a memory cycle, so that the address must be saved and then reissued. It is also conceivable that two requests could occur simultaneously, requiring that a decision

be made regarding which address will be issued first. Not only must the address be manipulated methodically, but the source or destination of the data associated with each address must be remembered by the control logic. These functions are accomplished for the most part by the central memory control logic, more often referred to as the Stunt Box.

Analysis of the Stunt Box occurs in the following sequence: hopper, priority network, and tag generation and distribution. These functions are described in detail on the following pages.

Hopper

The hopper is a mechanism used to save conflicting addresses so they may be reissued to the memory banks repeatedly (if necessary) until the addresses are accepted and processed. In addition to saving addresses, the hopper saves gating information used to enable the data corresponding to each address through the memory data distribution logic to or from memory.

Physically, the hopper is four flip-flop registers (designated M1, M2, M3, and M4) each of which stores an 18-bit address, 6-bit tag, and a Full bit (except M2, which has no Full bit). Refer to figure 77 at the end of this chapter. The registers are connected to each other in order to allow information to circulate through each of the registers (the concept is similar to the peripheral processor barrel). A 75ns time interval exists between each register and produces a total recirculation time of 300ns. For example, an address entered into M1 at time 00 enters M4 at t75, M3 at t150, M2 at t225 and, if it must be reissued, reenters M1 at t300.

In actuality, only 17 of the 18 available address bits are used in a 131K central memory. The 18th bit is found throughout the memory circuitry, but it is never utilized. The six-bit tag is generated when an address is first entered into the hopper (specifically, M1), and it contains all the information necessary to properly distribute the associated data. Tag generation and distribution is almost a subject in itself and is treated separately later in this section. The Full bits found in M1, M3, and M4 indicate that a meaningful address and tag are contained in the respective register. The Full bit is set when an address and tag are entered into M1. There is no Full bit in M2, since the Accept signal serves a similar purpose.

Approximately 50ns after entering an address into M1, the address is automatically sent to the memory banks where the lower five bits are examined to select one of the 32 banks. If the desired bank is not in use ($\overline{\text{Busy}}$), an Accept signal is returned to the Stunt Box to indicate that no conflict exists and the memory cycle has been initiated. The address saved in the hopper is then discarded. If the desired bank is busy, an Accept will not be returned. Its absence causes the associated address to be reentered into M1 (from M2) and subsequently reissued to the memory banks. The cycle will recur every 300ns until the address is accepted.

Figure 64, a timing diagram, verifies that the Accept signal is returned to the Stunt Box in time to disable (or if not returned, enable) the transfer of M2 to M1. If an Accept is generated, it will be received on chassis 5 about 175ns after entering the associated address into M1 (t175). This allows 125ns of logic delay time before the Accept is used to disable the M2 → M1 transfer (t300). Since M2 was not transferred to M1, the following transfer of M3 to M2 will destroy (write over) the contents of M2.

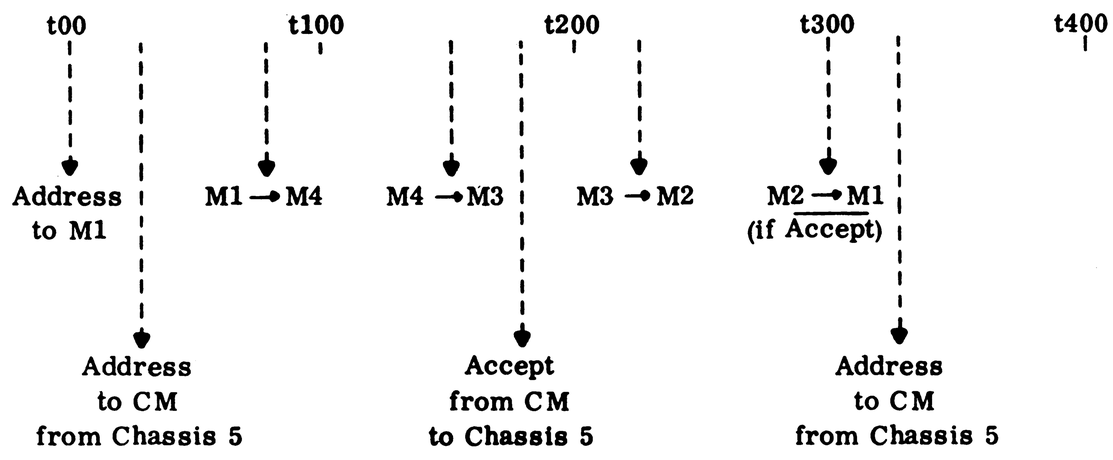


Figure 64. Timing for Accept Signal Return to Stunt Box

Priority Network

Because more than one source exists for addresses entered into M1 (figure 77) a stunt box priority network is necessary to ensure organized handling of simultaneous memory requests. Each address source has a fixed priority, as follows:

First	Hopper (M2 → M1)
Second	Central Processor (M0 → M1)
Third	Peripheral Processor (ERW → M1)

Hopper Priority. In only one circumstance is reentry of an address to M1 required. That occurs when an address sent to the memory banks was not accepted due to a bank conflict. Nonacceptance of an address is indicated by not receiving an Accept from the memory banks 175ns after issuing an address. Before enabling an M2 → M1 transfer, determine whether or not M2 contains a meaningful address. Check for the presence of a Full bit. Since M2 does not contain a Full bit, check the M3 Full bit (it is time delayed to ensure that M3 has been transferred to M2 before the check is made). Two conditions must be met to grant first priority: (M3 Full) ($\overline{\text{Accept}}$).

Central Priority. Two subpriorities exist under central priority because central processor memory references can be originated in two independent operations. These are: 1) instruction word fetching (RNI's) and 2) reading and storing operands.

In the first case, the address is obtained from the P register and in the second, from one of the two Increment functional units. If requests from both sources occur simultaneously, the operand address is entered first, then the instruction address. In either case, the address is entered into an 18-bit register, M0 (figure 77). At the same time, the enter central FF is set which indicates that an address is in M0 waiting for entry into M1. In a sense, the enter central FF requests priority 2. Thus, one condition required for priority 2 is that the enter central FF is set to indicate that an address is in M0 waiting for entry into M1.

A second condition needed for central priority is that priority 1 does not exist; i.e., the address in M2 was accepted or M2 does not contain a meaningful address.

A special circumstance arises which also must be considered in granting central priority. This is the case when read and store requests are made to the same memory address. This might occur when an instruction word modification is made followed by an RNI request for the modified word. If the two addresses enter the hopper in sequence (store location X, then read location X) storing before reading cannot be guaranteed because a bank conflict may exist with the store address. The operation (read or store) that is performed first depends strictly on when the bank becomes $\overline{\text{Busy}}$. Whichever address is sent to the banks first (after $\overline{\text{Busy}}$) will be accepted and will cause a conflict for the second reference to the same location. Thus, in the above instance, it would be possible to read the unmodified instruction word when actually the modified word was desired. The reverse situation might also occur, wherein a location was to be read before modification.

To resolve the above cases, additional logic is required in the priority 2 circuitry which prevents a central read address from being entered into the hopper if any (peripheral or central) write address is in the hopper. Also, if a central write is attempted, no central read address may be in the hopper. Prevention of a central write and peripheral read out of sequence is a software responsibility.

The fourth and final condition needed for central priority is that the address being referenced must not be out of bounds for its particular program. Memory bounds for a program are defined upon initialization of the routine (Exchange Jump) by the RA (reference address) and FL (field length) values. RA specifies the lower bound and RA + FL - 1 specifies the upper bound. Each central memory reference adds to the value RA, the content of P (for RNI's) or the increment 1 or 2 address (for operand references). The address being referenced (P, increment 1 or increment 2) is said to be the relative address. The absolute CM address is the sum of the relative address and the content of RA. The relative address is always entered into M0. A special adder adds M0 to RA to give the absolute address. Another circuit compares the content of M0 with the content of FL. If M0 $\geq$ FL, the desired reference is out of bounds and the memory reference will not take place because central priority will not be granted. Therefore, the condition M0 $<$ FL is also a condition required for granting priority 2.

The following Boolean formula summarizes the conditions required for granting central priority: (Enter Central) $\overline{\text{(Priority 1)}}$ (M0 $<$ FL) $\overline{\text{(Attempt Read) (Write in Hopper)}}$ + (Attempt Write) $\overline{\text{(Central Read in Hopper)}}$.

Peripheral Priority. Peripheral priority for CM references is granted only if neither hopper nor central priority exists and there is a PPU request for a CM access. Since only one PPU request can occur at a time, no subpriorities are required.

The PPU's request CM references in three situations. They are: 1) read central memory, 2) write central memory, and 3) exchange jump. In all three cases a PPU sends an 18-bit address to the Input Address register (IAR) of the Stunt Box (figure 77). To specify the type of reference being requested, a Read, Write, or Exchange signal pulse accompanies the address. These pulses are used as a control function to properly gate information to and from CM (see the hopper tag explanation which follows). The presence of one of the three control pulses results in the request for priority 3. Thus, the Boolean expression for peripheral priority is as follows: (PPU Read + Write + Exchange) $\overline{\text{(Priority 1)}}$ $\overline{\text{(Priority 2)}}$.

During peripheral processor read and write operations in central memory, a new address is sent to the IAR for every memory reference desired. For exchange jumps, only the starting address of the exchange jump package (in CM) is sent. It is the responsibility of the central processor to advance this address automatically in order to exchange the required information. This is accomplished by the exchange address counter (EAK) which is utilized only during exchange jumps. It increments the exchange address for each of the 16 locations referenced.

Hopper Tag Generation and Distribution

As previously mentioned, when an address is entered into M1, a six-bit tag is also entered. It is used

to properly gate data into and out of central memory. The tag bit positions are shown in figure 65.

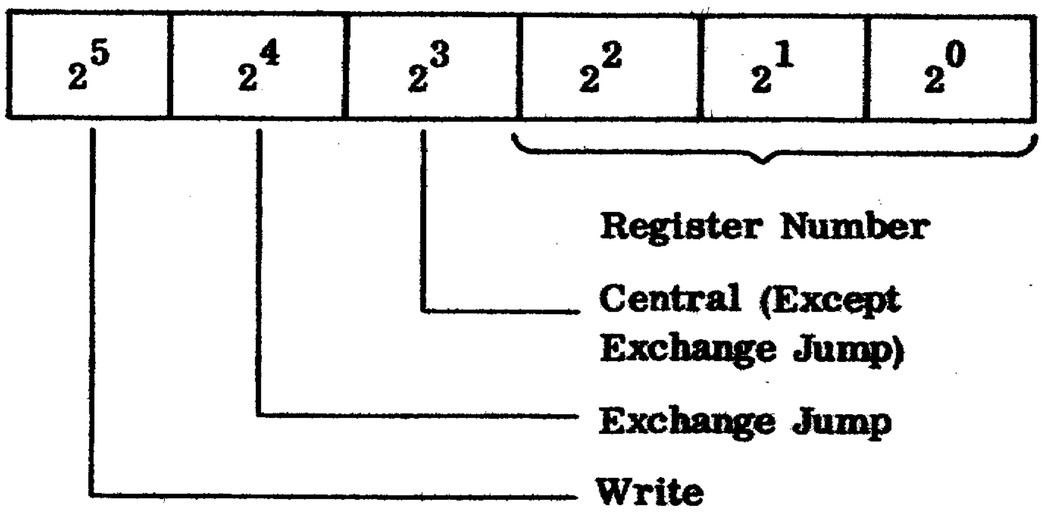


Figure 65. Tag Bit Positions

Bit (2^5) is set when the associated address is one of information to be stored (written) into central memory.

Bit (2^4) is set only during exchange jumps to indicate that the associated data is to be exchanged with registers in the CPU.

Bit (2^3) is set any time a memory reference is initiated by the central processor (priority 2). It allows information to be gated to or from the CPU, as opposed to a PPU. During exchange jumps the bit is set to indicate that an X register is to be exchanged, or it is cleared to indicate that A, B, and control registers (P, RA, FL, etc.) are to be exchanged.

Bits (2^0 through 2^2) indicate (when applicable) which X, B, or A register number is to be stored, read into, or exchanged. Table 11 lists all legal tag numbers (in octal) and their meaning. Decoding circuitry exists only for those tags listed. Any other bit combination is either not decoded, or is decoded as one of the legal tags.

A tag = 00 indicates a peripheral read address since all bits equal zero. This is interpreted as meaning: $\overline{\text{(Write)}}$ $\overline{\text{(Exchange)}}$ $\overline{\text{(Central)}}$ + Peripheral Read. In this case the register bits (2^0 through 2^2) have no meaning and are not translated.

A tag = 10 indicates a central RNI since the Central bit (2^3) is set and all other bits are cleared. Since a central read of memory to X0 is not possible, the clear state of bits 2^0 through 2^2 in this case indicates that an instruction word is to be read from memory.

A tag = 11 indicates that a central read to X1 is to be performed. Bits 2^0 through 2^2 in this case indicate the X register number. Tags 12 through 15 are also central reads to X registers, but to X2 through X5, respectively.

A tag = 40 indicates a peripheral write operation, since the Write Bit (2^5) is set and the Central and Exchange bits (2^3 and 2^4) are both cleared.

A tag = 50 indicates a central return jump or error mode stop memory reference. Bits 2^0 through 2^2 are meaningless in this case since a central write (tag = 5X) of X0 is not possible. Since storage of information in central memory is required in the above cases, the 50 tag is reserved for this purpose.

Tags = 56 and 57 are generated when storage of X6 or X7 is desired. Bits 2^0 through 2^2 again indicate the register number.

Tags 60 through 77 are all generated during an exchange jump operation. Bit $2^3 = 0$ indicates that A, B, or control registers are to be exchanged. Bit $2^3 = 1$ indicates that an X register is to be exchanged. Bit 2^0 through 2^2 specify the operating register number (X, B, or A) or the control register (P, RA, FL, or EM) to be exchanged. Note that these are the only cases when bit 2^4 (Exchange bit) is set.

After a memory reference is initiated, the associated tag is decoded, and it will enable the gating of the desired information into and/or out of central memory, to or from the desired location (XBA registers, read or write pyramids, etc.).

Table 11. Hopper Tag Definitions

Number	Meaning
00	Peripheral Read
10	CP RNI
11	CP Read → X1
12	CP Read → X2
13	CP Read → X3
14	CP Read → X4
15	CP Read → X5
40	Peripheral Write
50	Return Jump + Error Stop
56	CP Write X6
57	CP Write X7
60	Exchange P, A0
61	Exchange RA(cm) A1, B1
62	Exchange FL(cm) A2, B2
63	Exchange EM, A3, B3
64	Exchange RA(ecs) A4, B4
65	Exchange FL(ecs) A5, B5
66	Exchange A6, B6
67	Exchange A7, B7
70	Exchange X0
71	Exchange X1
72	Exchange X2
73	Exchange X3
74	Exchange X4
75	Exchange X5
76	Exchange X6
77	Exchange X7

CENTRAL MEMORY

The 6600 central memory is composed of 60-bit words located in 16 or 32 memory banks each of which contains 4K words. This results in 65K or 131K memory sizes, respectively. In either case four banks are contained on a chassis.

Selections of bank and chassis are made by decoding the lower four (for 65K memories) or five bits (for 131K memories) of the address. For example, in a 131K system bits 2^0 and 2^1 select one of four banks on a chassis, while bits 2^2, 2^3, and 2^4 select one of eight chassis. The address is sent from chassis 5 to all memory chassis of a system and all chassis decode the lower bits of the address. Only one chassis will recognize its bit configuration (2^2 through 2^4). By decoding bits 2^0 and 2^1, the bank selection is made. If the selected bank is free (a memory cycle is not already in progress) the Accept signal is returned to the Stunt Box and a Go signal is set to the selected storage sequence control circuit (SSC). The SSC is a simple flip-flop timing chain which generates the read/write memory cycle.

The selected address within the selected bank is determined by decoding the remaining 12 bits of the 17-bit address (16 bits for a 65K system). While a memory cycle is in progress, the Bank Busy signal (Bank Not Free) disables initiation of other memory cycles within that bank. It also disables the return of the Accept signal to chassis 5 which causes the address to be retried at the 300ns stunt box rate.

The information being read from or stored into central memory is gated by a circuit called the read/write distributor. It distributes information to and from the four chassis connected to central memory as shown in figure 66.

Determination of which information is to be gated is made by decoding the hopper tag associated with each address and ANDing the decoded signal with the Accept for that address to determine which information should be gated.

- Information is gated from the Write Pyramid on chassis 1 to CM via the write distributor during PPU central memory write operations (tag 40).
- Information is gated from chassis 5 during return jumps and error mode stops (tag 50).
- Information is gated from chassis 7 and 8 during exchange jumps (tags 60 through 77) and central processor store operand operations (tags 56 or 57).
- Information is sent from CM via the read distributor to chassis 1 during PPU central memory read operations (tag 00).
- Information is sent to chassis 5 during exchange jumps (tags 60 through 65) and RNI (tag 10) operations.
- Information is sent to chassis 7 and 8 during exchange jumps (tags 60 through 77) and central processor operand read (tags 11 through 15) operations.

It is reemphasized that in all cases fo gating the read/write distributor the Accept signal is necessary. This ensures that the information desired is properly timed for entry to or exit from memory.

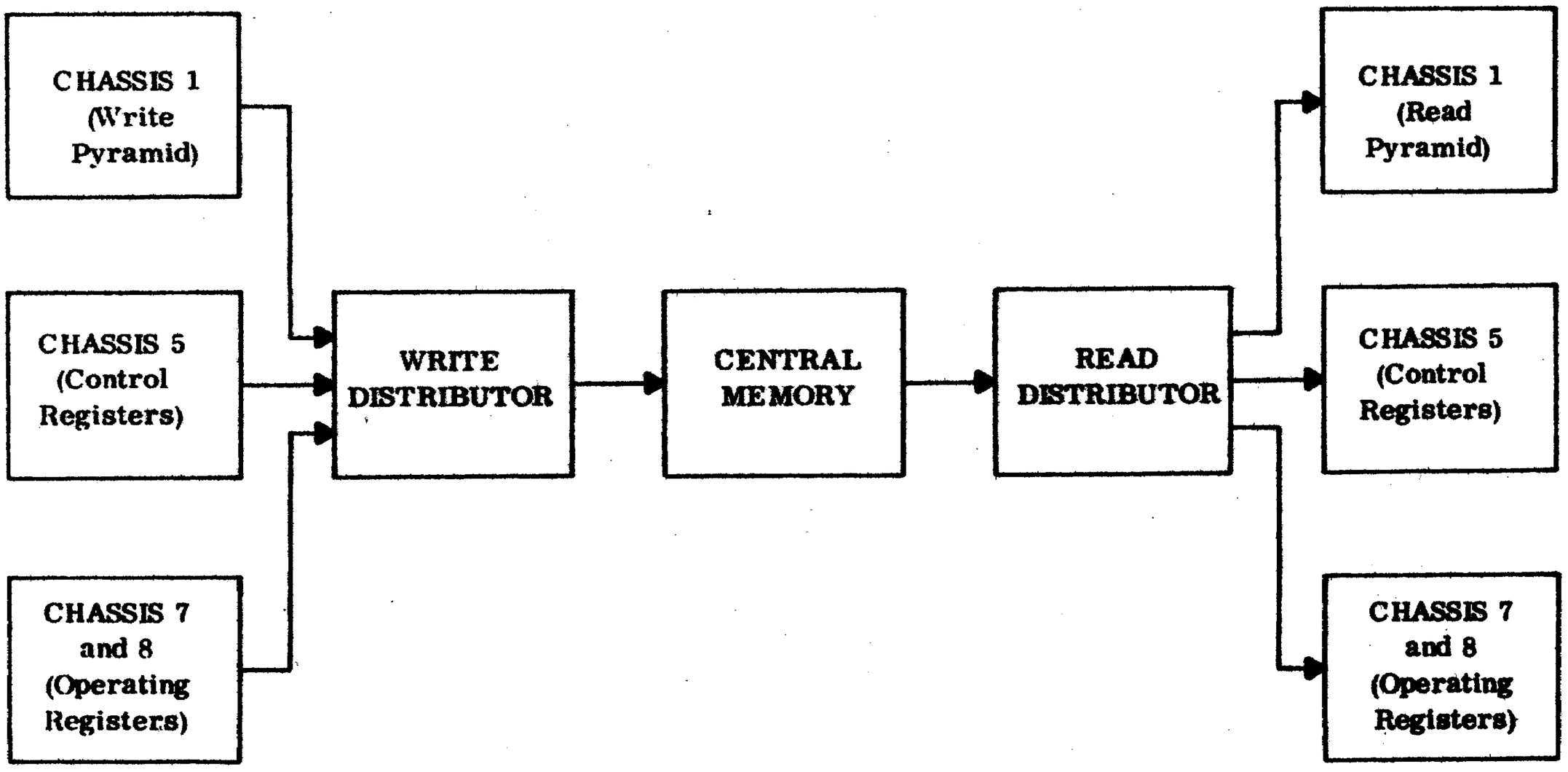


Figure 66. Read/Write Distributor

INSTRUCTION CONTROL

Instructions in the central processor are executed from the Instruction Stack (figure 75). Each 60-bit Stack register can contain up to four instructions, but not less than two, since the CPU employs both a 15-bit and a 30-bit instruction format. The responsibility of Instruction Control is to: determine that a 60-bit instruction word has become available to the stack, sort out the 15- or 30-bit instructions within that word, and then deliver the instructions to Reservation Control so they can be executed.

Initially all instruction words (a 60-bit memory word fetched by an RNI request) move into the bottom rank of the Stack (I0) when Instruction Control receives the RNI tag (tag = 10_8 and Accept) from the Stunt Box. The RNI tag also signals Instruction Control to begin the process of sorting instructions within that word and transfering them to Reservation Control. The total process is called instruction issue, and the sorting of instructions is referred to as parceling.

INSTRUCTION ISSUE

Instructions can be issued from any rank of the Stack. However, if we assume an initial condition such as at the end of an exchange jump sequence, we can see issue beginning with the upper instruction in I0. Once a program is in execution, program control can be transferred to some higher rank of the Stack by a Branch instruction. This situation forces Instruction Control to keep track of which rank of the Stack the program address is currently indicating. The Locator (L) register and counter perform this function. Control of the L count would be very similar to control of the program address. However, L refers to a particular rank of the Stack so it can only vary between 0_8 through 7_8. For example, L count = 0_8 indicates program control is currently in I0. L count = 7_8 indicates program control in I7. An initial Master Clear sets the L count to 0_8, so we can see that instruction issue would start from the bottom rank of the Stack.

NOTE

The L register contains the complement of the L count.

Parceling

Now that we have selected a particular rank of the Stack, we must concern ourselves with sorting out the instructions within that rank. In other words we must parcel the instructions from the selected rank of the Stack. Each rank is considered to have four overlapping 30-bit parcels. These are: parcel 0 (bits 30 through 59), parcel 1 (bits 15 through 44), parcel 2 (bits 00 through 29), and parcel 3 (bits 44 through 14, end around).

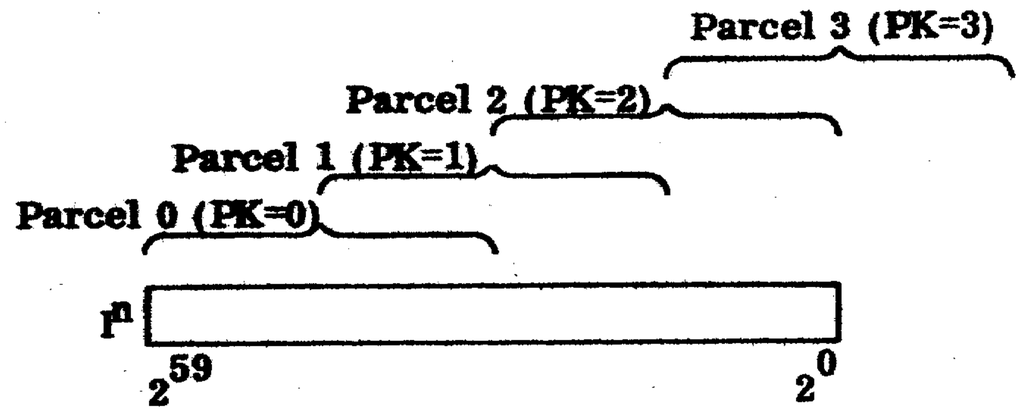
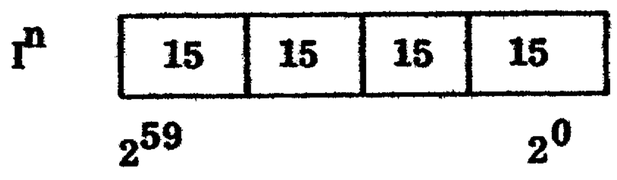


Figure 67. Instruction Word Parcels

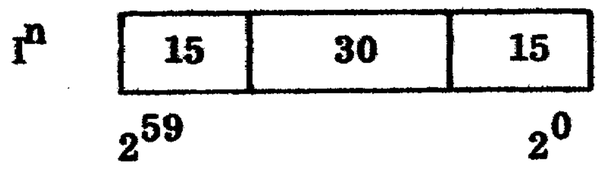
By examining each word with the 30-bit parcels we can guarantee the detection of any possible combination of instructions within that word. If the instruction contained within the parcel happens to be a 15-bit instruction, the lower 15 bits of the parcel are discarded and the next sequential parcel is extracted. However, if a 30-bit instruction is encountered, the entire parcel would be used and the next sequential parcel would be skipped.

Example 21



In example 21 each parcel would be extracted in sequence with the lower 15 bits of each parcel being discarded.

Example 22



In example 22 parcels 0 and 1 would be extracted in sequence. All 30 bits of parcel 1 would be used, so parcel 2 would be skipped.

Instruction Control uses a two-bit parcel counter (PK) to keep track of the current parcel. It generally advances the parcel counter after each parcel is extracted. An initial PK=0 condition can be set by a Master Clear.

Initial instruction issue begins with the L count = 0 (I0), PK=0 (bits 30 through 59). And, as previously mentioned, the first instruction word after an exchange jump comes from central memory to I0 as a result of the RNI tag. The tag also starts the issue operation. The rest of our analysis is concerned with moving the instructions to Reservation Control. Figure 68 shows the path each parcel takes.

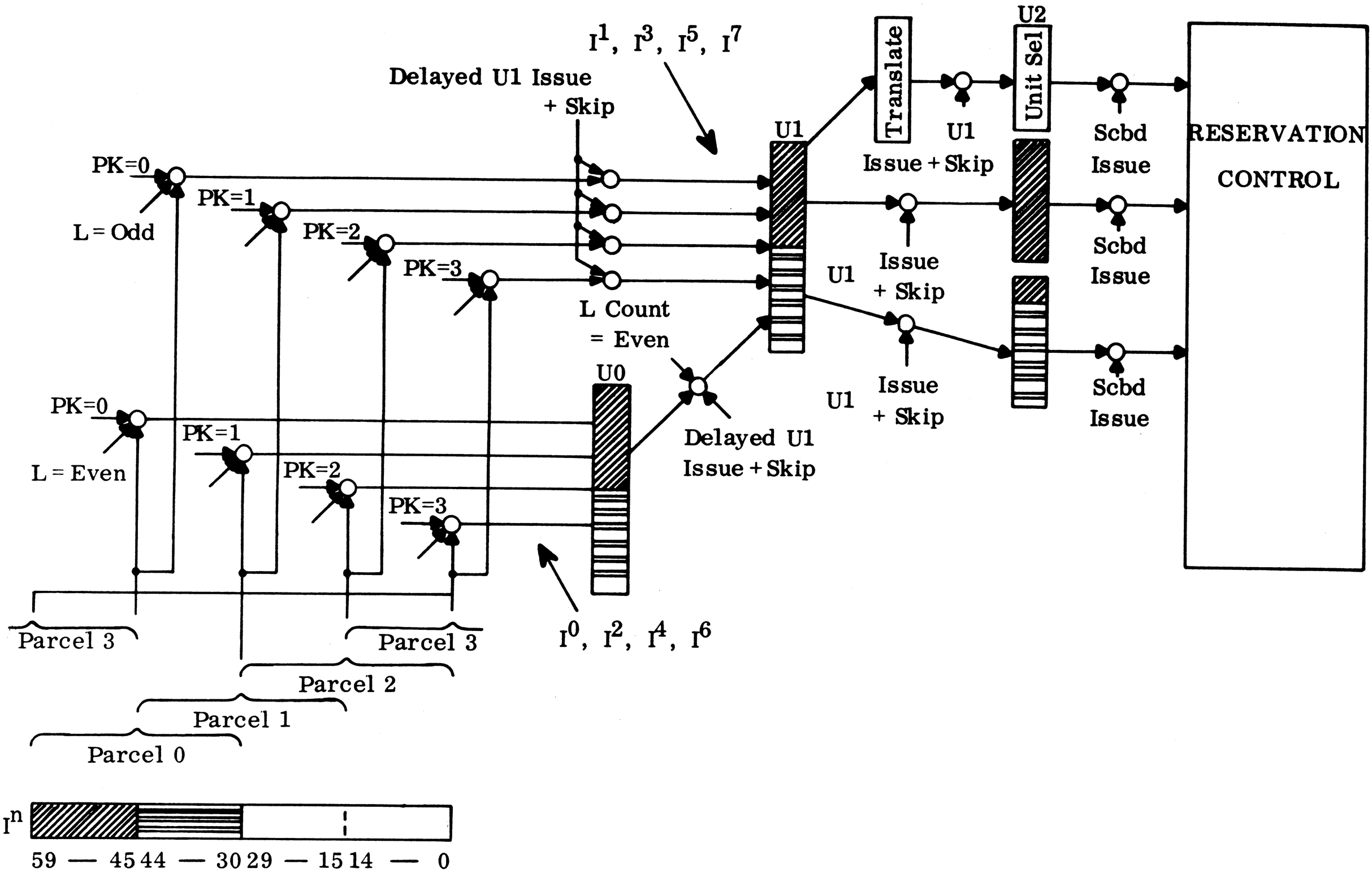


Figure 68. Parcel Counter

ISSUE CONTROL

Issue Control generates two types of issue pulses. These are U1 issue and scoreboard issue. U1 issue is a pulse that gates the selected parcel to the U1 and U2 instruction registers and advances PK. This pulse occurs at a minor cycle rate during the issue sequence. Scoreboard issue is a pulse that gates the parcel from the U2 instruction register to Reservation Control. This issue can also occur at a minor cycle rate.

With eight different L counts and four different parcel counts, it is easy to see that 32_{10} different parcels must move through the U1 and U2 registers. Sixteen of these parcels (PK = 0, 1, 2, 3; L count = 0, 2, 4, 6) move from the even-numbered ranks of the Stack to U0 before U1 issue moves them to U1. It is not necessary to have an Issue pulse to move parcels to U0. The selected parcel moves from the selected even rank of the Stack into U0 automatically. In our initial case parcel 0 of I0 is the first parcel extracted to U0 and the first U1 Issue pulse moves the parcel to U1.

Notice that at this time there have not been meaningful parcels in U1 or U2. So as far as the U2 register is concerned it receives trash on the first U1 issue. Also, no scoreboard issue should be generated until after the first meaningful parcel has moved into U2. Because PK is advanced to one by U1 issue, parcel 1 of I0 is extracted to U0 so on the next U1 issue parcel 1 would move to U1, parcel 0 would enter U2, and PK would advance to two.

One more U1 issue moves parcel 2 to U1 and parcel 1 from U1 to U2. At this time the first scoreboard issue must occur to issue parcel 0 to Reservation Control. From this point both U1 and scoreboard issue can continue at a minor cycle rate until parcel 3 is issued to Reservation Control (three more Issue pulses).

All of the possible instructions in I0 are now in execution and issue must stop until the next 60-bit instruction word becomes available from central memory. This stop is called a pause.

Pause

The Pause can be quite lengthy if Instruction Control does not have the foresight to request another RNI from central memory. This request is made any time L count = 0, PK = 0, and U1 issue. It is easy to see why the request is made under these conditions once it is realized that I0 is the bottom rank of the Stack and after issuing from I0 there isn't any place to go for the next instruction. There is one other operation that comes into play at this time. It is the process of moving the current instruction words of the Stack up to make room for the next instruction word from central memory (inching).

Inching

Inch is also started when L count = 0, PK = 0, and U1 issue. During the inch process each rank of the Stack is shifted up starting with an I6 to I7 transfer and continuing with I5 to I6, I4 to I5, I3 to I4, I2 to I3, I1 to I2, and I0 to I1 in that sequence.

Four minor cycles (400 ns) are necessary to complete inch. It is necessary to advance the L count to one since the last inch transfer moves the current instruction word from I0 (L count = 0) to I1 (L count = 1). It is important to realize that the I6 to I7 transfer destroys the instruction word that previously was in I7. Consequently, a program loop that is to be executed within the Stack must fit in the Stack between I1 and I7. A quick examination of the Stack reveals a maximum in-stack program length of 27_{10} instructions (table 12).

Table 12. Maximum In-Stack Loop

I7	15	15	15	15
I6	15	15	15	15
I5	15	15	15	15
I4	15	15	15	15
I3	15	15	15	15
I2	15	15	15	15
I1	15	15	30-Bit Branch	
I0				

The analysis of instruction issue to this point has assumed straight line program execution with no complications. However, there are many special situations which may be encountered. Whenever a 30-bit instruction is encountered in a parcel, Instruction Control must cause the next sequential parcel to be skipped. The skip is accomplished by not generating a scoreboard issue when the unwanted parcel is in U2 (figure 69).

There is also the possibility that issue may have to be stopped if either the functional unit or the result register (required by the parcel) are busy. This is accomplished by Instruction Control translating the parcel when it is in U1 and setting a unit request FF and result register select FF when the parcel enters U2. Each unit request FF interrogates the corre-

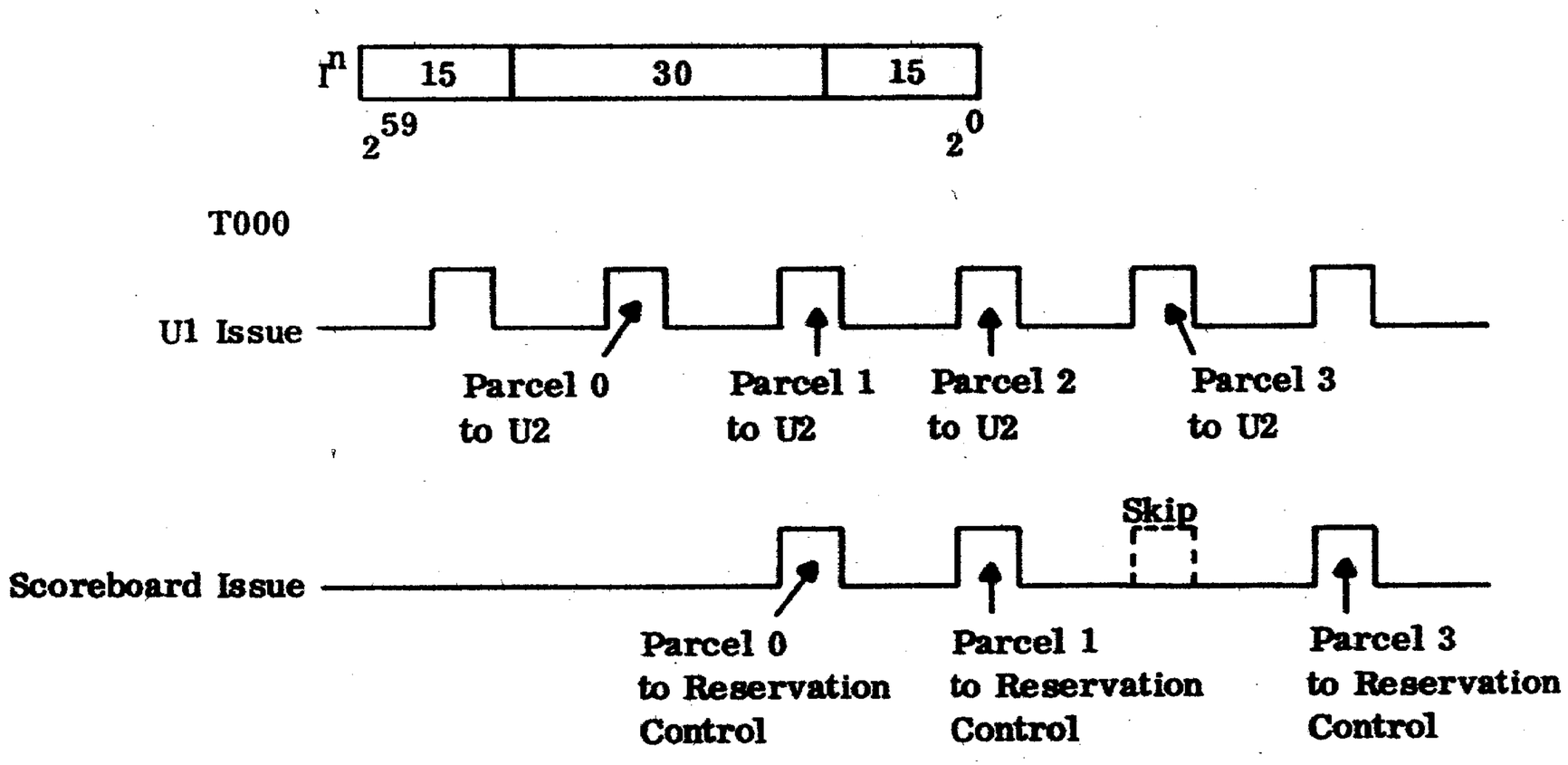


Figure 69. Skip Generation

sponding unit busy FF in Reservation Control, and, if the unit is busy, a signal is generated which blocks all Issue pulses until the unit becomes not busy.

A similar operation occurs with the result register except the result register select FF must be ANDed with a translation for the i portion of the parcel to determine which portion of the reservation list should be examined for a reservation. The reservation list (XBA) is where all result register reservations are held by Reservation Control.

Probably the most involved operation in Instruction Control occurs when a Branch instruction is encountered. And even though the Branch unit will execute the instruction, Issue Control must set itself up for proper operation. The reason for the complication is that a Branch instruction can do one of the following three things:

- Loop if a conditional branch, condition met, and in the Stack.
- Jump if an unconditional branch or a conditional branch, condition met, and not in the Stack.
- No branch if a conditional branch and condition is not met.

Instruction Control sets itself up for the no branch condition by setting the parcel count equal to the parcel count of the next parcel to be issued. It then stops issue after issuing the branch (figure 70).

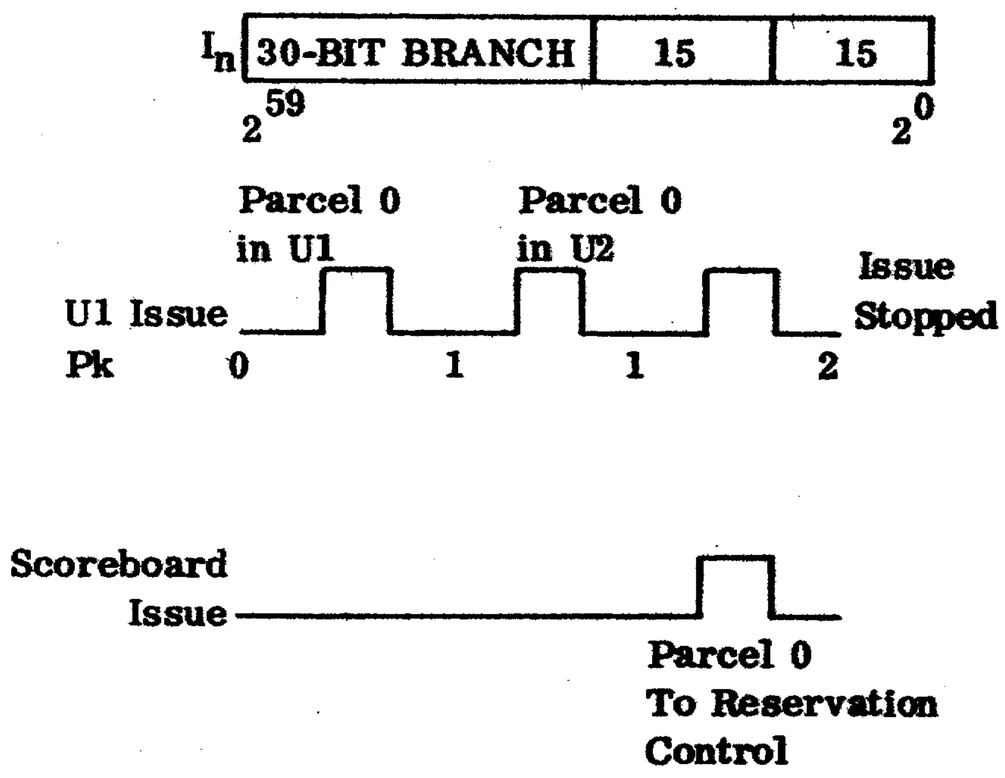


Figure 70. No Branch Condition

By controlling the parcel count, it is possible to restart issue in the same manner as if it were starting after an RNI. Once the branch has been issued, the Branch unit makes a series of tests to determine if the branch is to an instruction word already in the Stack.

The test results are only enabled on the conditional branch instructions 03_8 through 07_8. The results also enable these instructions to loop. The first test is made by subtracting the current program address (P) from the jump address (R). If the difference (T) is ±7 or less, the branch may be in the Stack.

However, a further test must be made to see if a jump of T places can be made relative to the current position in the Stack which is reflected by L. This is the L-T test and, if there is not an end-around borrow from the test, the branch still may be in the Stack.

If R-P gives a positive result, the jump is forward and the L-T test being made successfully stays in the Stack. But if R-P is negative, the branch is backward and a further test must be made to see if there is a usable instruction in the rank of the Stack to which the jump is being made. Conveniently, the result of the L-T test is the new L setting if the branch is made. This is subtracted from the Stack Depth Counter register (D). The D-(L-T) test is only necessary on the backward jumps (R-P negative). If the jump is successfully made, the branch is in the Stack. The Branch unit uses the Long Add unit to make the branch condition test for the 03_8 instructions and the Increment units to test the 04_8 through 07_8 instructions. If the condition is met, a Go Branch signal is generated. If the R-P, L-T, D-(L-T) are all successful, a Loop Proceed is generated.

If Go Branch occurs and R-P, L-T, and D-(L-7) were not successful, a jump is generated. If Go Branch does not occur, a No Branch Proceed is generated. On a Loop Proceed, the jump address (R) is transferred to P, the result of the L-T test is gated to L, the parcel count is set to 0_8, and issue is restarted. On the jump an R to P transfer is also accomplished, but L is set to 7_8 (L count = 0_8), the parcel count is cleared, and an RNI request is made to the Stunt Box. Issue restarts as a result of the RNI. The No Branch Proceed merely restarts issue, since this is why Instruction Control has been set up. There are many special cases that affect Instruction Control during branch instructions, but these will be covered in chapter 3 of the 6600 Central Processor Training Manual, Volume 1, along with a more detailed explanation of the other Instruction Control operations.

RESERVATION CONTROL (SCOREBOARD, FIGURE 79)

The need for reservation control logic in the 6600 CPU arises because of the parallel processing concept of the CPU. This capability necessitates an orderly means of utilizing the functional units, operating registers, and memory circuitry since it is possible that

several instructions may require the same functional unit, operating register, etc. The Scoreboard makes the required reservations for each instruction and provides a means for the orderly handling of conflicts which can occur between instructions.

Conflicts are categorized into three orders: first, second, and third. The type of conflicts are defined on the following pages.

- FIRST ORDER. A conflict between two instructions that require the same functional unit or the same result registers is a first-order conflict.

Example 23 Functional Unit Conflict

```
F X 6 = X1 ⊕ X2
F X 5 = X3 ⊕ X4
```

Both instructions need the Floating Add functional unit for their calculation. Since only one such unit exists, the second instruction must wait until the first has finished before it can be executed. Note that if two Multiply instructions are coded in sequence, no functional unit conflict occurs since two Multiply units are provided.

Example 24 Result Register Conflict

```
F X6 = X1 + X2
F X6 = X4 * X5
```

Both instructions require X6 for their result. In this case, the floating add result would be returned to X6 before the multiply result was desired.

There are two types of first-order conflicts: functional unit and result register. In all cases of first-order conflicts, issuance of instructions stops until the conflict is resolved. In other words, no further instructions are initiated (including the one which sees the conflict) until the first of the conflicting instructions has completed execution. First-order conflicts temporarily stop issuance of instructions at the point of conflict.

- SECOND ORDER. A conflict that occurs when an instruction requires the result register of a previously initiated instruction as a source operand is a second-order conflict.

Example 25

```
F (X6) = X1 + X2
F X7 = X5 / (X6)
```

In example 25, the Divide unit needs X6, which is the result of the Add instruction, as one of its source operands. The Divide unit must obviously wait for the Add unit to time out, but instruction issue does not stop. Instead, the Scoreboard delays the start of the Divide instruction until the Add unit has stored its result. Subsequent instructions may be issued as long as no first-order conflicts exist. The result of a second-order conflict is to delay the execution of the conflicting instruction only.

- THIRD ORDER. A conflict that occurs when one instruction must store its result in a register which is to be used as a source operand for a previously issued instruction is a third-order conflict.

Example 26

```
F X3 = X1 / X2
F X5 = (X4) * X3
F (X4) = X0 + X6
```

In example 26, due to the relatively long execution times of the divide and multiply op codes and the second-order conflict (X3) of these units, the Add instruction completes its calculation before the Multiply unit reads its operands. Both operands are always read at the same time. Therefore, all second-order conflicts must be resolved. Since the Multiply instruction is intended to read X4 before it is changed by the Add instruction, storage of the add result must be delayed until the Multiply unit begins its calculation. Thus, where third-order conflicts do not delay issue or calculation, they do delay the storage of a result operand.

The following discussion explains, at the block diagram level, how these conflicts are handled by the Scoreboard. Figure 71 should be used in following the explanation.

FIRST-ORDER CONFLICTS

First-order conflicts are defined as either functional unit or result register conflicts. If either type exists, no issues can be generated. Functional unit conflicts are determined by checking the unit request FFs against the unit busy FFs. Recall that unit requests are set at the time an instruction is transferred from U1 to U2. The unit request FF sets depending upon the op code that is translated (from U1 fmi portion). When an instruction is issued to the Scoreboard a unit busy FF sets. This, in effect, reserves a functional unit for a particular op code. It will remain reserved until the given instruction has completed execution. Then the unit busy FF is cleared to allow a subsequent instruction to use that functional unit.

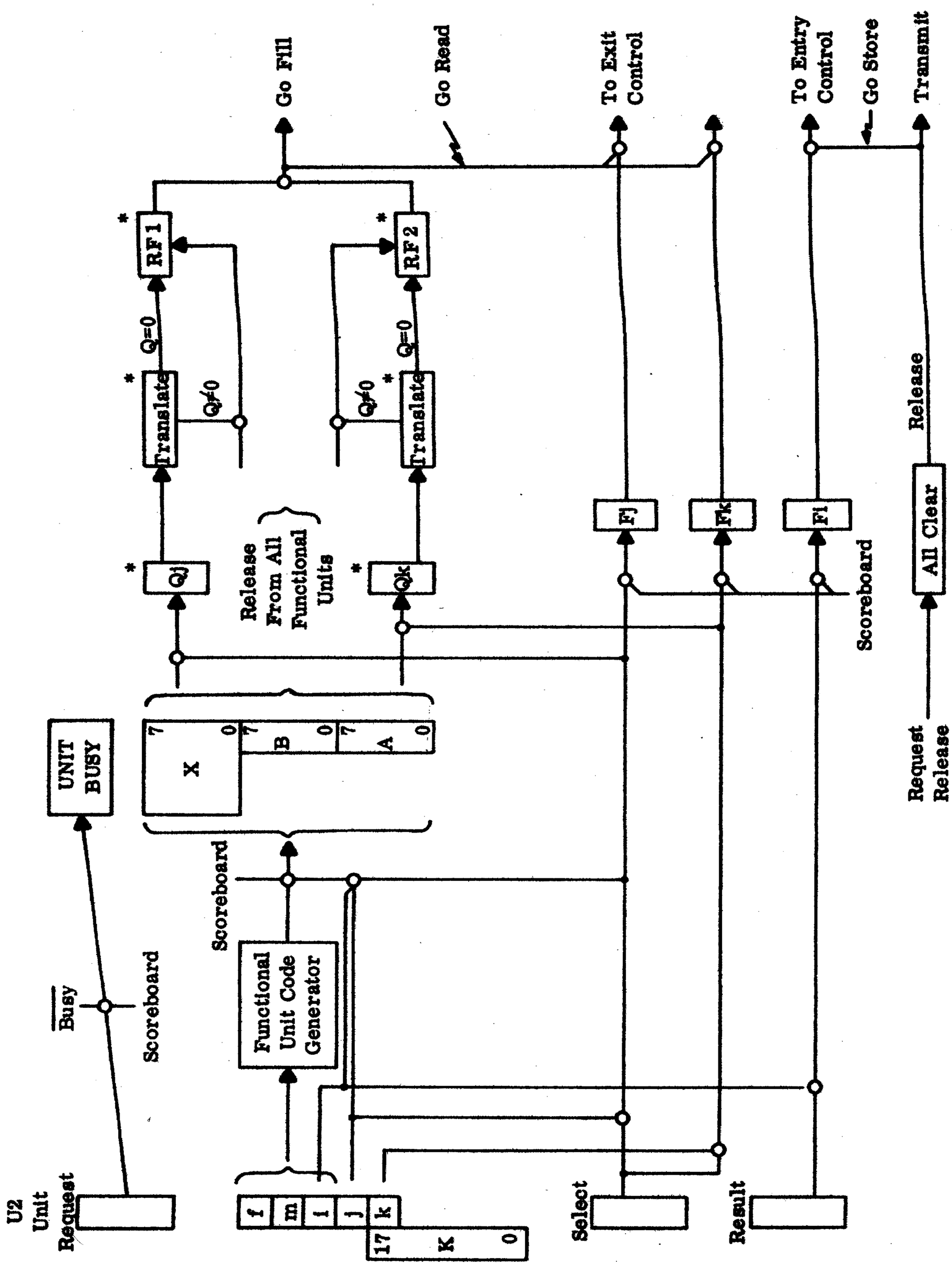


Figure 71. Scoreboard Block Diagram

So when a given unit is needed (as determined by a set unit request FF) it can be used only if the associated unit busy FF is cleared. If the unit busy FF is set, a functional unit conflict exists and generation of issues is disabled until the unit is freed.

In determining the existance of result register conflicts, a comparison of the request and reservation logic is also made. Upon issuing an instruction to U2, result FFs are set according to the op code translation. Four such FFs exist and specify a result register group (i.e., Xi, Bi, Ai or Bj). The specific register within a group is determinde by translating the i or j octals (as specified by the result FFs). For example, if the Xi FF is set and the U2 i digit = 3, X3 is the result register desired.

The result register reservations are placed in the XBA reservation list when an instruction is issued to the Scoreboard. This list is composed of 24 slots, where codes are placed to specify which functional unit has reserved each of the 24 operating registers. For example, a code of 16_8 in the X4 slot of the reservation list indicates that X4 is reserved for the result of the Long Add unit. The complete list of possible codes appears in table 13.

Table 13. Functional Unit Codes

Unit	Code
Increment 1	01
Increment 2	02
Shift	03
Boolean	04
Divide	05
Multiply 1	06
Multiply 2	07
Read Memory, Channel 1	11
Read Memory, Channel 2	12
Read Memory, Channel 3	13
Read Memory, Channel 4	14
Read Memory, Channel 5	15
Long Add	16
Add	17

Any slot that contains a code of all zeros indicates that the associated operating register is not reserved. Any nonzero code indicates that the associated register is reserved for a result. If translation of U2 indicates that X4 is specified as a result register and the X4 slot of the reservation list is zero, no conflict occurs. If the X4 slot is not equal to zero, the register is reserved. A conflict exists and issues are disabled until the conflict is resolved.

Notice from the list of codes that five are named Read Memory, Channel X. These are necessary for the 5 X 1 through 5 X 5 instructions. They return results from memory to X1 through X5 and must make a result register reservation. In this sense, memory acts like a functional unit.

In summary, both cases of first-order conflicts are handled similarly in that requests for units or result registers (made at U2 time) are checked against reservations existing in the Scoreboard. If a conflict exists, issues are disabled until the conflict is resolved.

SECOND-ORDER CONFLICTS

Second-order conflicts occur when a functional unit requires as a source operand the result of another functional unit. The source operands are defined by the j and k octals of U2 in conjunction with the select FFs which are set with a U2 transfer. The select FFs define the source register group as well as the octal digit specifying the register within that group (i.e., Xj, Bj, Aj, Bk or Xk). Specific registers are selected by ANDing select FFs with the j and k octal digit translations.

Determination of whether or not the desired registers are reserved is made by indirectly looking at the content of the XBA reservation list. Each functional unit has four bit Q designators which, when an instruction is issued to the Scoreboard, receive the contents of the XBA slot associated with the desired source operand registers. For example, the following instruction sequence causes a second-order conflict.

Example 27

```
F X 5  =  X3 * X2
F X 6  =  X2 + X5
```

The Multiply 1 unit reserves X5 by placing a code of 06_8 in the X5 slot of the reservation list. Assuming that no other instructions have been issued, no other reservations exist when the Add instruction is issued. Since the Add unit wished to read X2 and X5, it transfers to its Qj and Qk designators the content of the X2 and X5 slots, respectively. At this point, the Add Q designator equals 00_8 and Qk equals 06_8. In essence, this tells the Add unit that its j operand (X2) is not reserved by the Multiply 1 unit. Since a functional unit does not begin calculation until after it can read both operands, the Add unit must wait until Multiply 1 returns its result to X5.

Associated with each functional unit are flip-flops, called read flags which, when set, indicate that the desired operand(s) can be read. The Add unit has two read flags, one for the Xj operand and one for Xk. Read flags can be set in two ways, both of which result from translating the Q designators.

- If $Q = 00_8$, a read flag can be set since the desired operand is not reserved.

- If $Q \neq 00_8$, a read flag cannot be set until the functional unit, whose code is in Q, has completed its calculation and returned its result to the result register. Completion of a functional unit's operation is indicated by a signal called Release (see description of third-order conflicts).

In example 27, the Add unit's Xj read flag is set immediately because Qj = 0. The Xk read flag is set when the Release for Multiply 1 occurs because Qk translates as 06_8. Each possible nonzero Q translation is tied to the Release signal for the associated functional unit. It is possible to set a read flag by any translation of Q ANDed with the associated Release signal or by $Q = 00_8$.

Once both read flags are set, it is necessary to send the functional unit its operands and to send a Go signal to the unit to allow it to begin its calculation. The Go FU signal is sent as soon as both read flags are set. This signal starts the functional unit timing chain. At the same time, the source register selection codes are sent (by a Go Read signal) to Register Exit Control to gate the proper operands to the unit. These codes are obtained from the F designators associated with each unit. These are three-bit designators which are used to remember the source and result operand register numbers. They also are set when the Scoreboard is issuing an instruction. In example 27, once both read flags are set, the contents of the Fj and Fk designators of the Add unit are sent to Register Exit Control and allow X2 and X5 to be gated to the Add unit. The read flags are cleared during the minor cycle after both are set. Set read flags indicate that an operand is waiting to be read.

The general second-order conflict case delays the start of a functional unit until both source operands can be read. Some special cases exist which are discussed in detail in the logic analysis sections of this manual. At this point, it is necessary only to understand the general case.

THIRD-ORDER CONFLICTS

The possibility of third-order conflicts occurs when a functional unit has generated a result and wishes to store it in an operating register. If the desired result is waiting to be read, the unit must wait to store until after the read has occurred.

Whether or not a register is waiting to read is determined by checking the Fj and Fk designators against the associated read flags in all the functional units. The result register of a unit is given by the Fi (and, in some cases, Fj) designator of that unit. When a unit requests to store a result, its result register number is checked against the read flags and F designators of all other units. If any read flag is set and the associated Fj or Fk designator translation is the same as the Fi designator of the storing unit, a third-order conflict exists. The unit is prevented from storing until the read flag of the conflicting unit is cleared. This occurs once a unit has set both of its read flags.

The general sequence in handling third-order conflicts is as follows. First, a unit desiring to store a result sends a Request Release signal to the Scoreboard near the end of its calculate time. This signal is then ANDed with an All Clear signal to generate the release gate, which allows storage of the result. The All Clear is the result of checking all read flags with the associated Fj and Fk designators and comparing them with the Fi (or Fj) designator (for the result) of the unit requesting release. The Release signal performs several necessary tasks in the Scoreboard. It sends a Transmit signal to the functional unit to gate the result to the data trunk. It also generates a Go Store signal which gates the Fi (or Fj) designator to Register Entry Control to select the desired result register. Release also clears reservations in the Scoreboard (i.e., XBA designators, unit busy FFs, etc). It also checks all Q designator translations in the event that a unit is waiting to use this result as a source operand. The Release indicates final termination of an instruction and removes that instruction from the Scoreboard.

REGISTER EXIT/ENTRY CONTROL

Register Exit/Entry Control is the control logic used for gating data into and out of the 24 operating registers. It is a large translating network which decodes tags sent from the Stunt Box or Scoreboard to enable the transfer of data to and from central memory or the functional units. Figure 72 is a block diagram which should be referred to while reading the following explanation.

ENTRY CONTROL

Entry Control is shown on the left half of figure 72. To the extreme left are the four general sources of information for the X, B, and A registers. They are central memory, data trunk 1 (Shift, Add, Long Add), data trunk 2 (Boolean, Divide, Multiply 1, Multiply 2), and data trunk 3 (Increment 1, Increment 2).

Data is entered into the operating registers from central memory during Exchange Jumps and during the Central Read Operand instructions (5 X 1 through 5 X 5). Since memory references are involved, all of the gating tags are sent from the Stunt Box and are composed of the lower four bits (2^0 through 2^3) of the hopper tag ANDed with the Accept signal for the associated address. When an exhcnage jump or a read operand address is accepted, the four-bit tag is sent

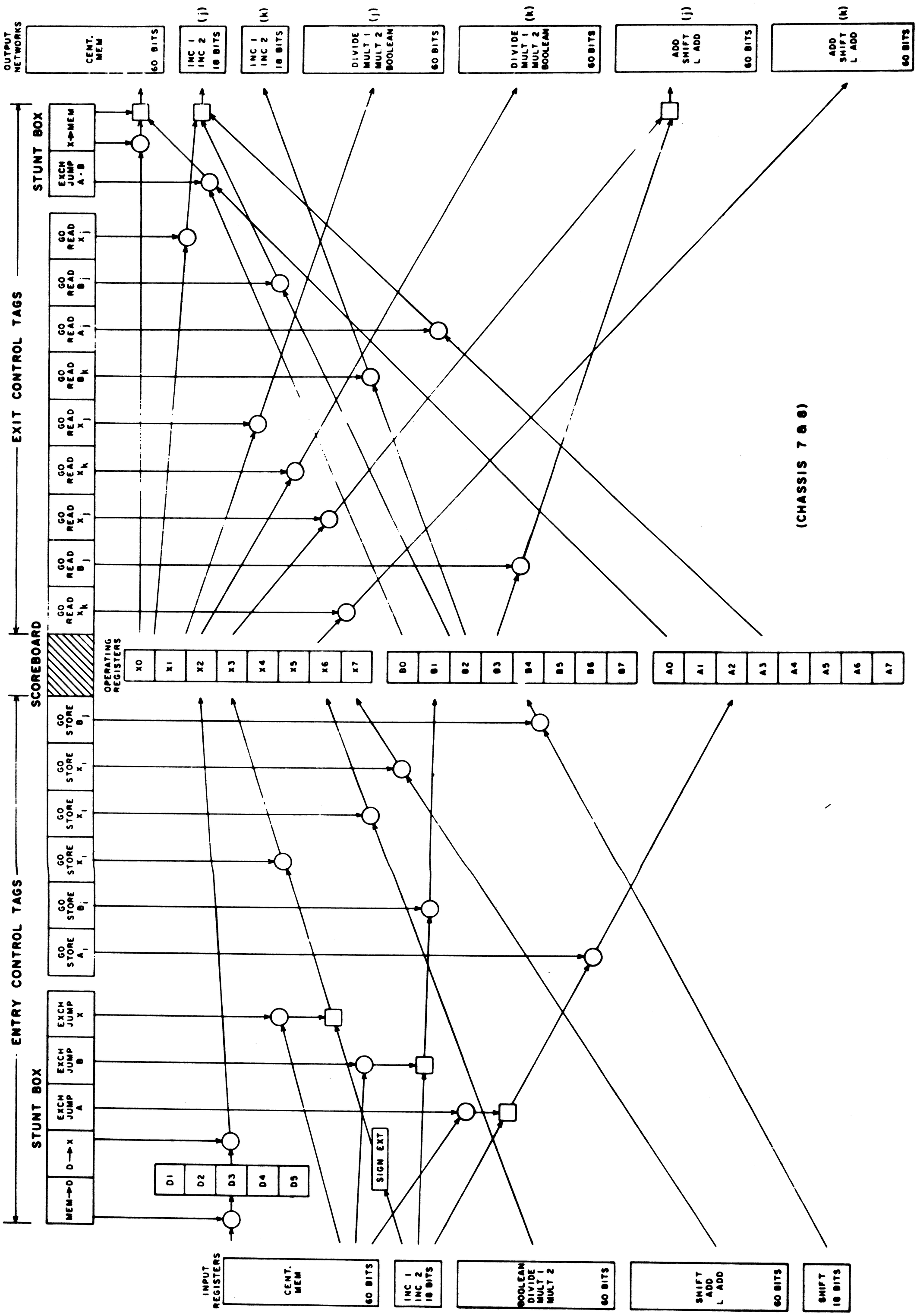


Figure 72. Register Exit/Entry Control Block Diagram

from the tag timing chain to Entry Control where it enables the information from central memory to the proper X, B, or A register.

Notice that during read operand references, the 60-bit operand is first sent into the D register (1 through 5) associated with the X register (1 through 5) which will ultimately receive the information. This transfer is enabled by the Memory → D signal which results from the simple translation: (tags 11 through 15)(Accept). The operand is temporarily stored in D until any third-order conflict which may exist is resolved. A read flag can be set for the X register which is to receive the operand from memory. When the All Clear signal occurs, the D → X signal is generated and completes the transfer to X.

During Exchange Jumps, new information is entered into the A, B, and X registers by hopper tags in the range, 60 through 77. Tags 60 through 77 enable the exchange of A and B registers and tags 70 through 77 enable exchanging X registers.

A result generated by the Increment units may be entered into X, B, or A registers, depending upon the instruction being processed (5X, 6X, or 7X). Therefore, three entry control tags are shown for the increment data trunk. They are go store Ai, go store Bi, and go store Xi. Recall that Go Store occurs after a functional unit has been released and enables the Fi designator content to Entry Control. The go store tags are generated by the Scoreboard at the completion of an instruction sequence. Note also that sign extension occurs when storing an increment result (18 bits) in an X register (60 bits).

Results generated by the Boolean, Divide, Multiply 1 or Multiply 2 functional units are always 60 bits in length, and the result register of these functional units is always an X register. One entry control tag, go store Xi, is shown for data trunk 2. The tag is also generated from the Fi designators of the units on this trunk when the unit is released by the Scoreboard.

The units on data trunk 1 (Shift, Add, and Long Add) all generate a 60-bit result for X registers. In addition, the Shift unit can generate an 18-bit result for a Bj register (during normalize or unpack operations). Therefore, two entry control tags are shown for this trunk. They are go store Xi and go store Bj. These tags are also generated by the Scoreboard when a unit releases from the Fi (or Fj in the special shift case) designators.

EXIT CONTROL

Like Entry Control, Exit Control has four general destinations for data from the operating registers. These are central memory, data trunk 1, data trunk 2, and data trunk 3. The exit control tags are also generated from the Stunt Box or the Scoreboard.

The Stunt Box generates tags for information which is sent to central memory during Exchange Jumps or Central Store Operand instructions (5 X 6 or 5 X 7). During these operations, the lower four bits of the hopper tag are sent to Exit Control when the associated address has been accepted (is not in conflict). The data, which can be A and B register or X register contents, is sent on the memory trunk and is stored during the write portion of the memory cycle.

For the increment data trunk, four exit control tags are shown in figure 73 since the Increment units can specify an A, B, or X register with the j octal and only a B register with the k octal. The four tags, go read Xj, go read Bj, go read Aj, and go read Bk are used to gate operands on this trunk.

For data trunk 2, all functional units specify only X registers as source operands. Therefore, only go read Xj and go read Xk tags are required.

For trunk 1, the Add and Long Add units can specify an Xj or an Xk source register (or both) while the Shift unit can specify an Xj or Bk register (or both). The three tags, go read Xj, go read Bj and go read Xk are required for this data trunk.

All go read tags are generated by the Scoreboard when both read flags for a unit have been set. This enables the Fj or Fk designator to Exit Control and gates the proper register to the proper trunk.

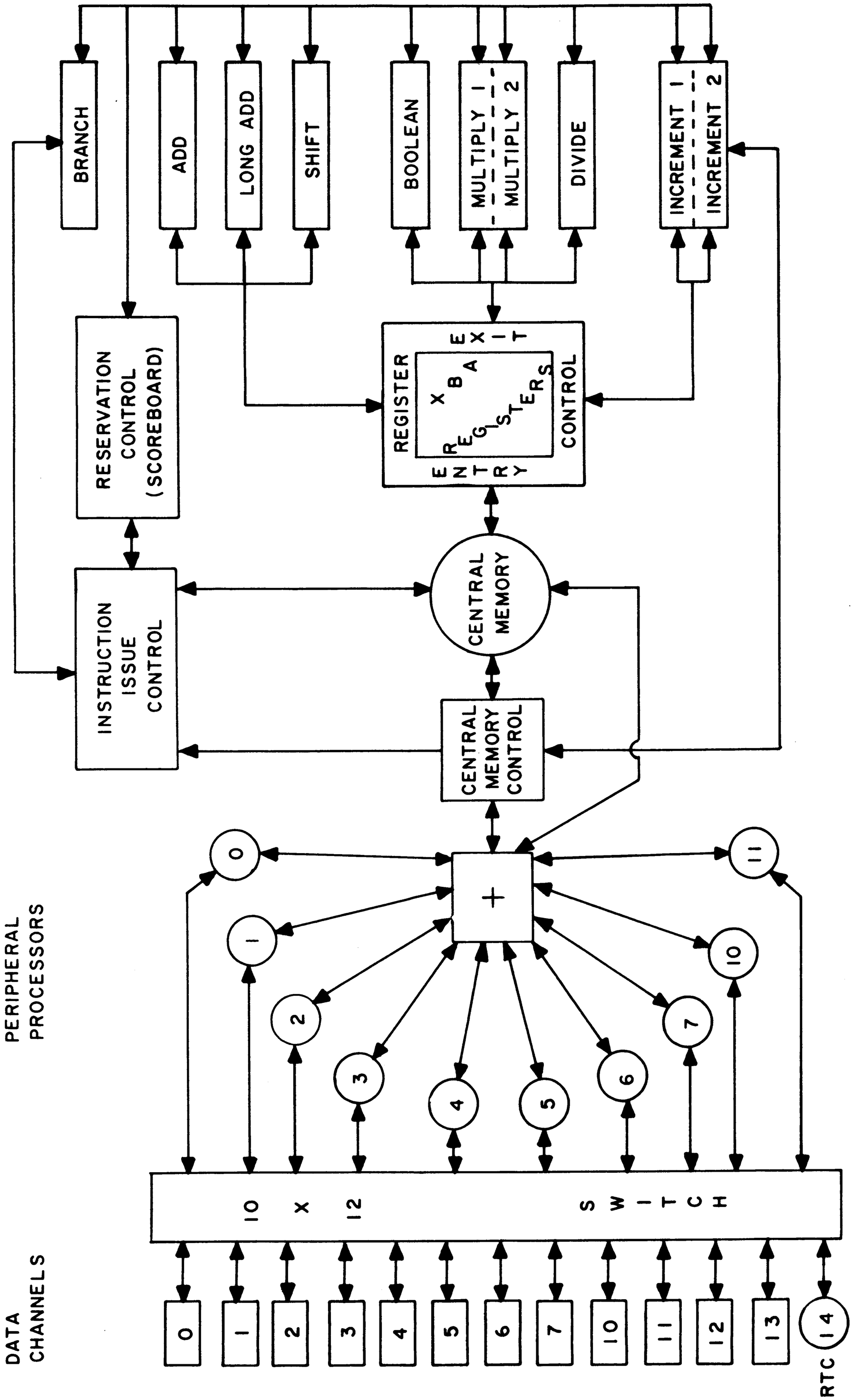


Figure 73. A General and Overall 6600 Computer Block Diagram

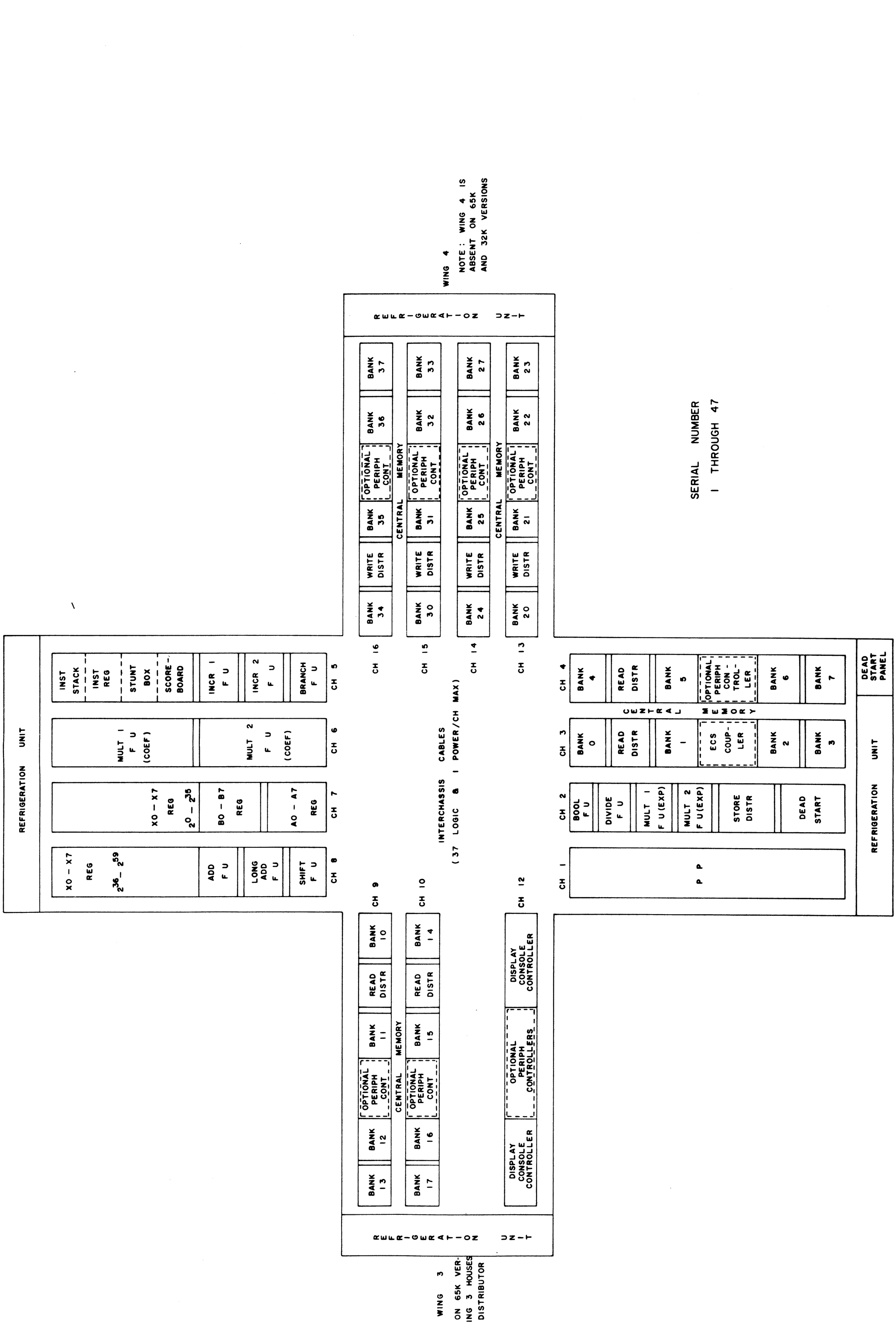


Figure 74. Central Processor Chassis Layout 131K

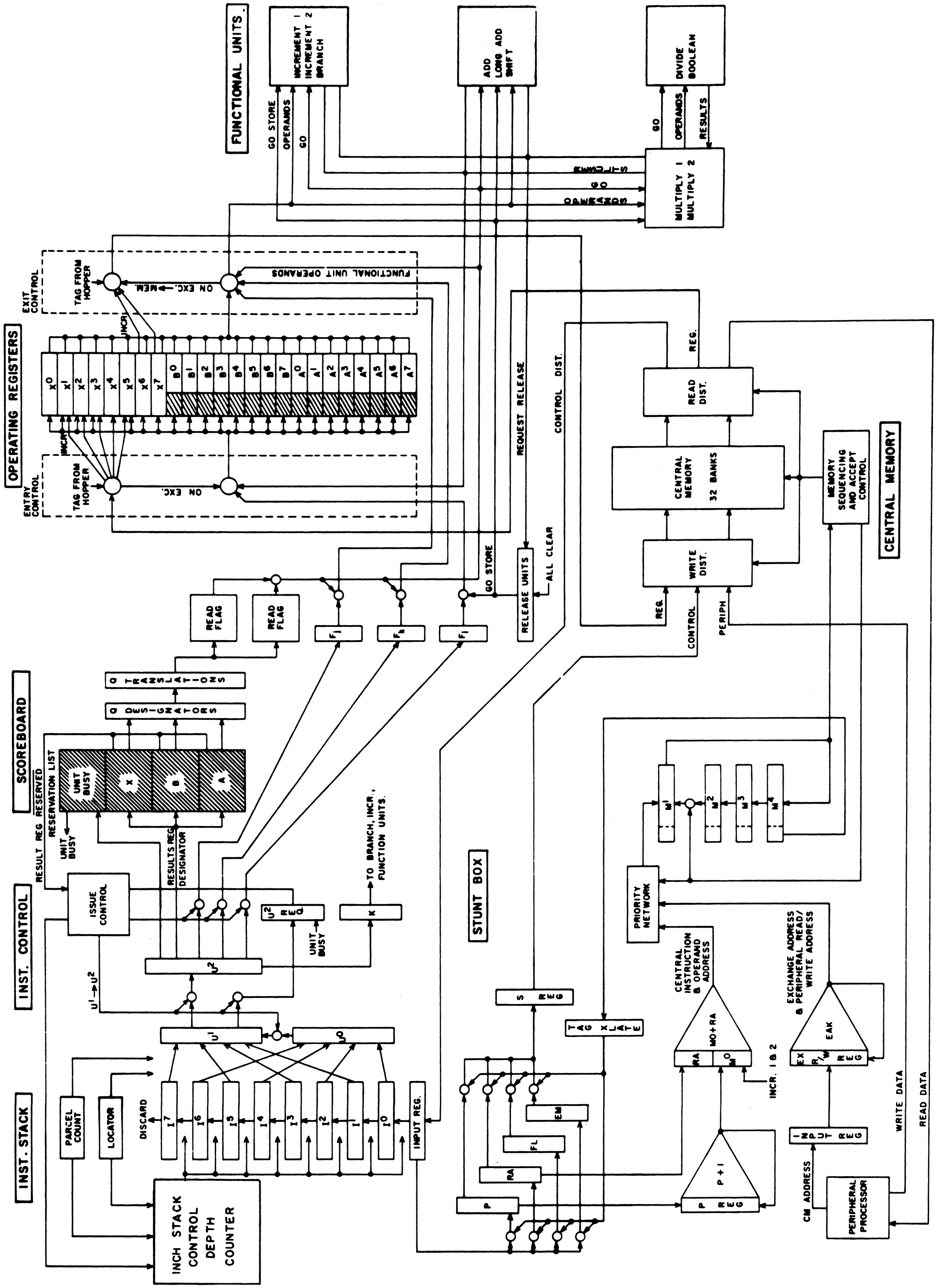


Figure 75. Central Processor Block Diagram

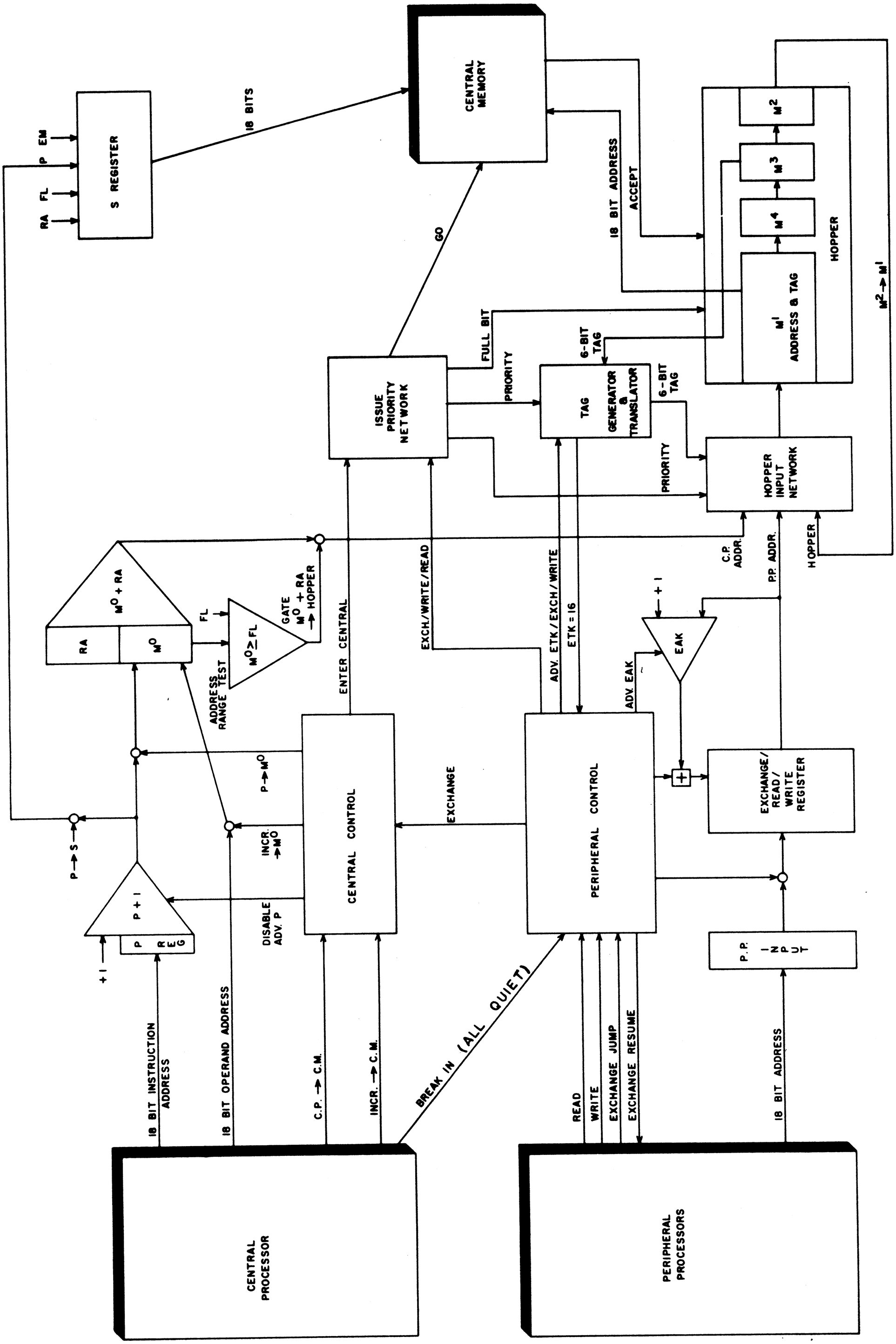


Figure 76. Central Processor Stunt Box Block Diagram

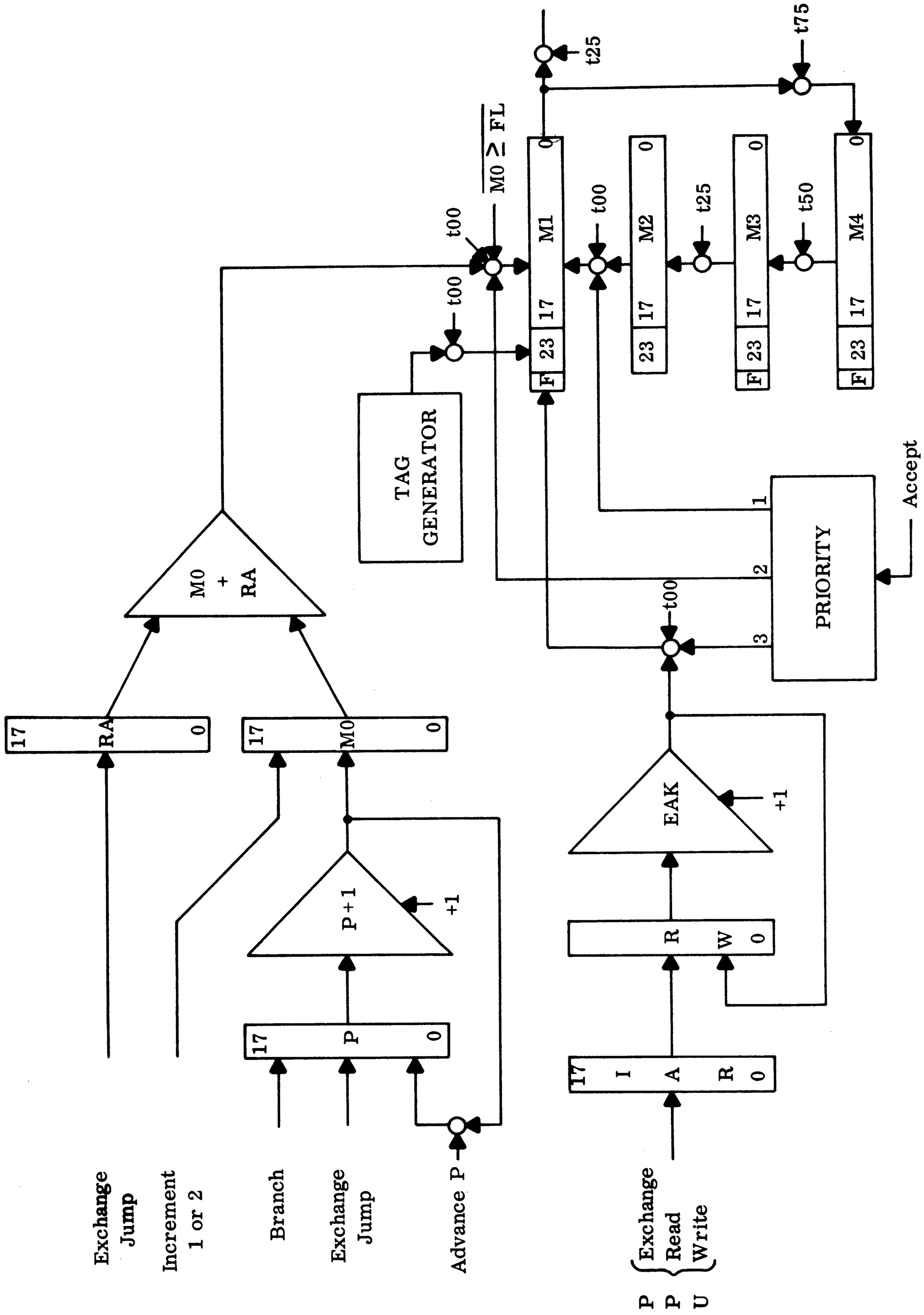


Figure 77. 6600 Stunt Box Detail Block Diagram

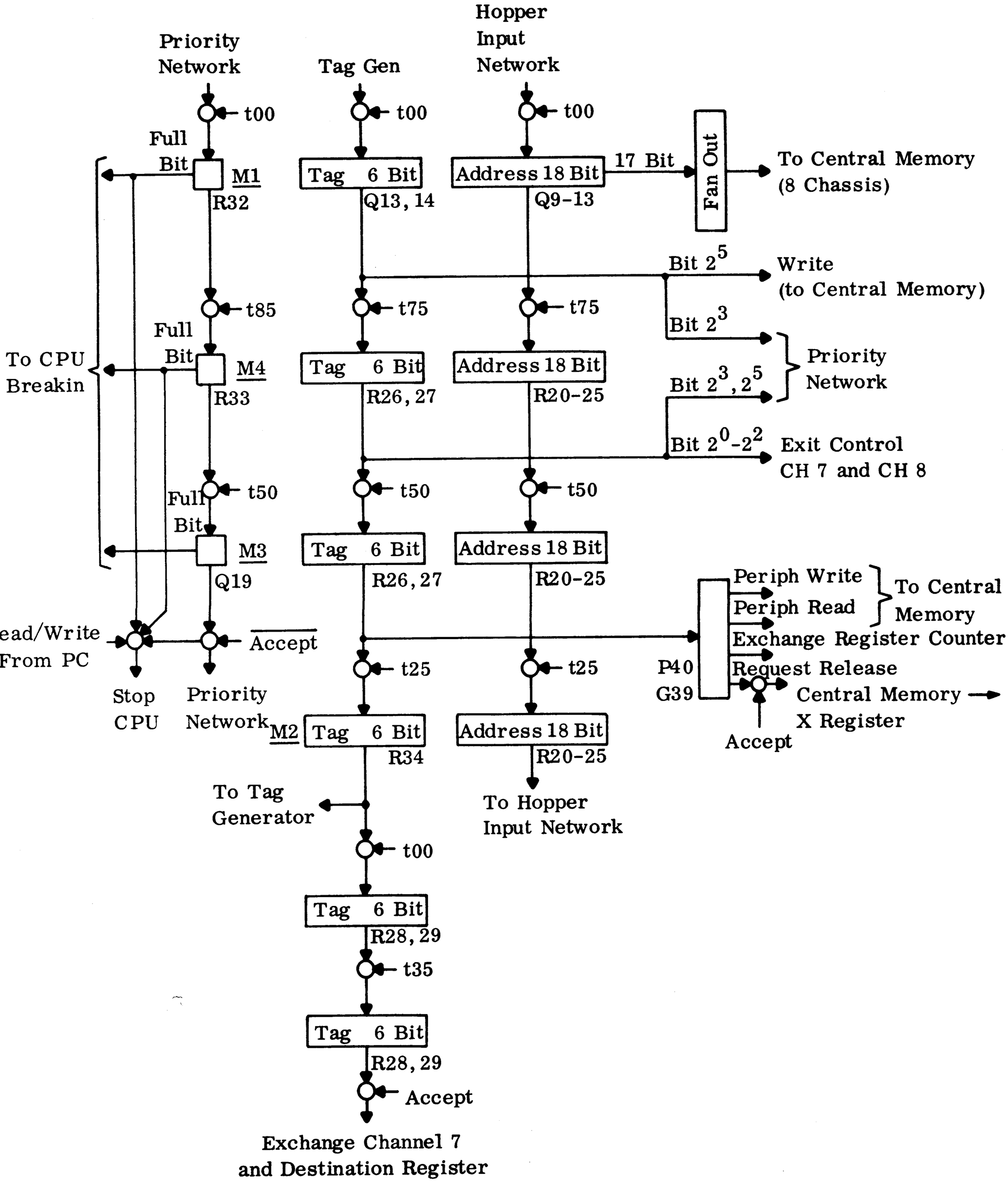


Figure 78. Hopper

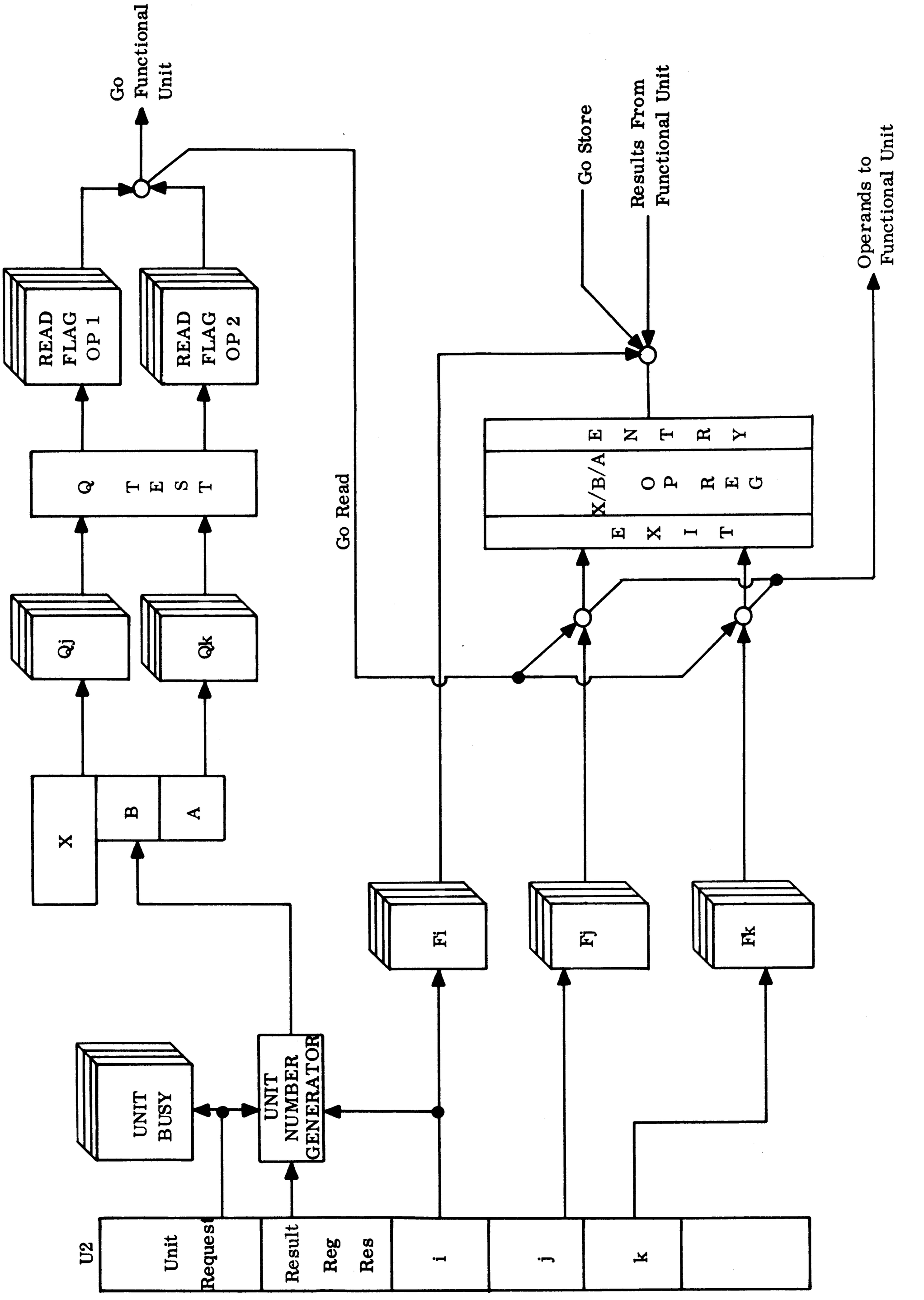


Figure 79. Scoreboard

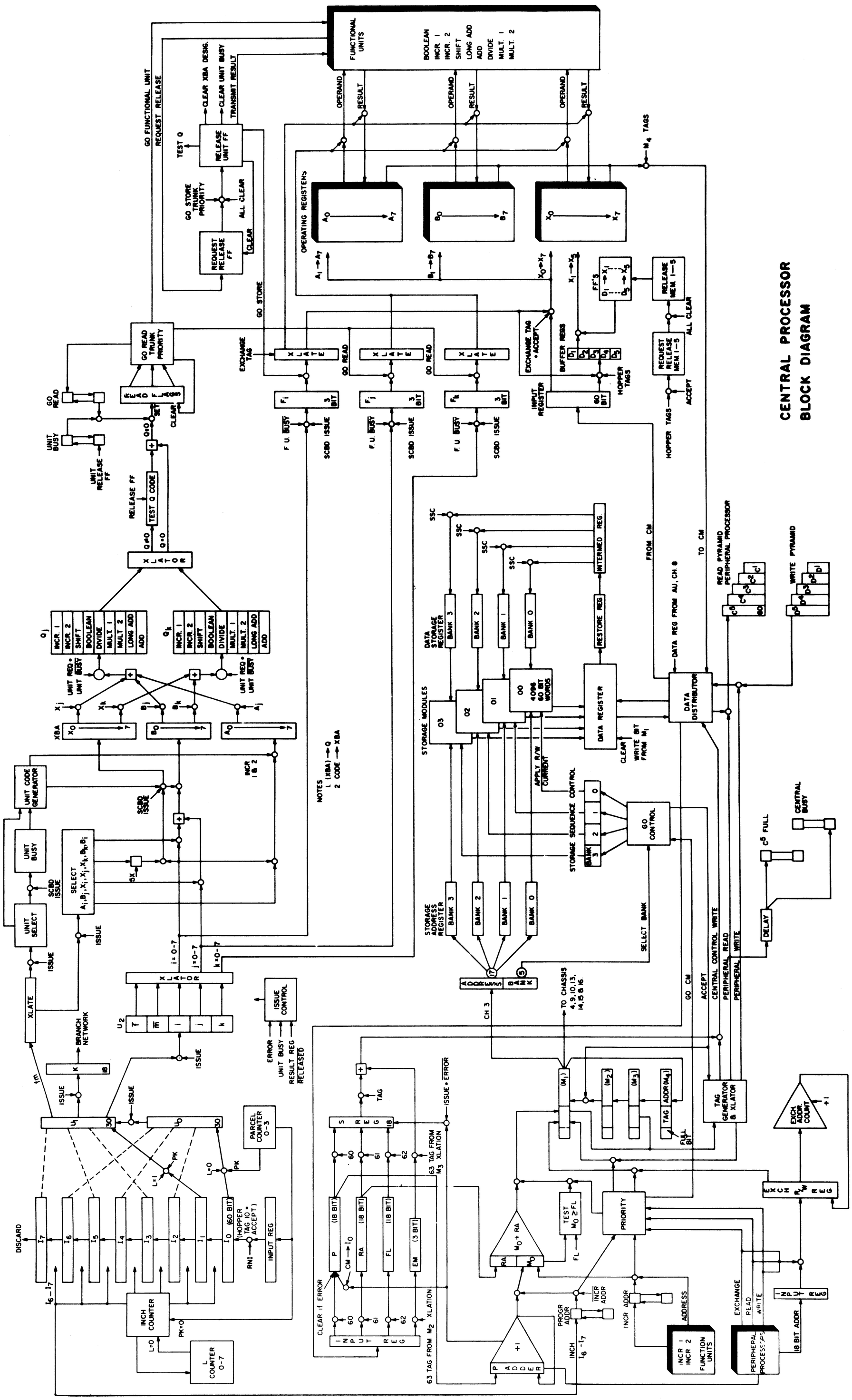


Figure 80. Central Processor Block Diagram

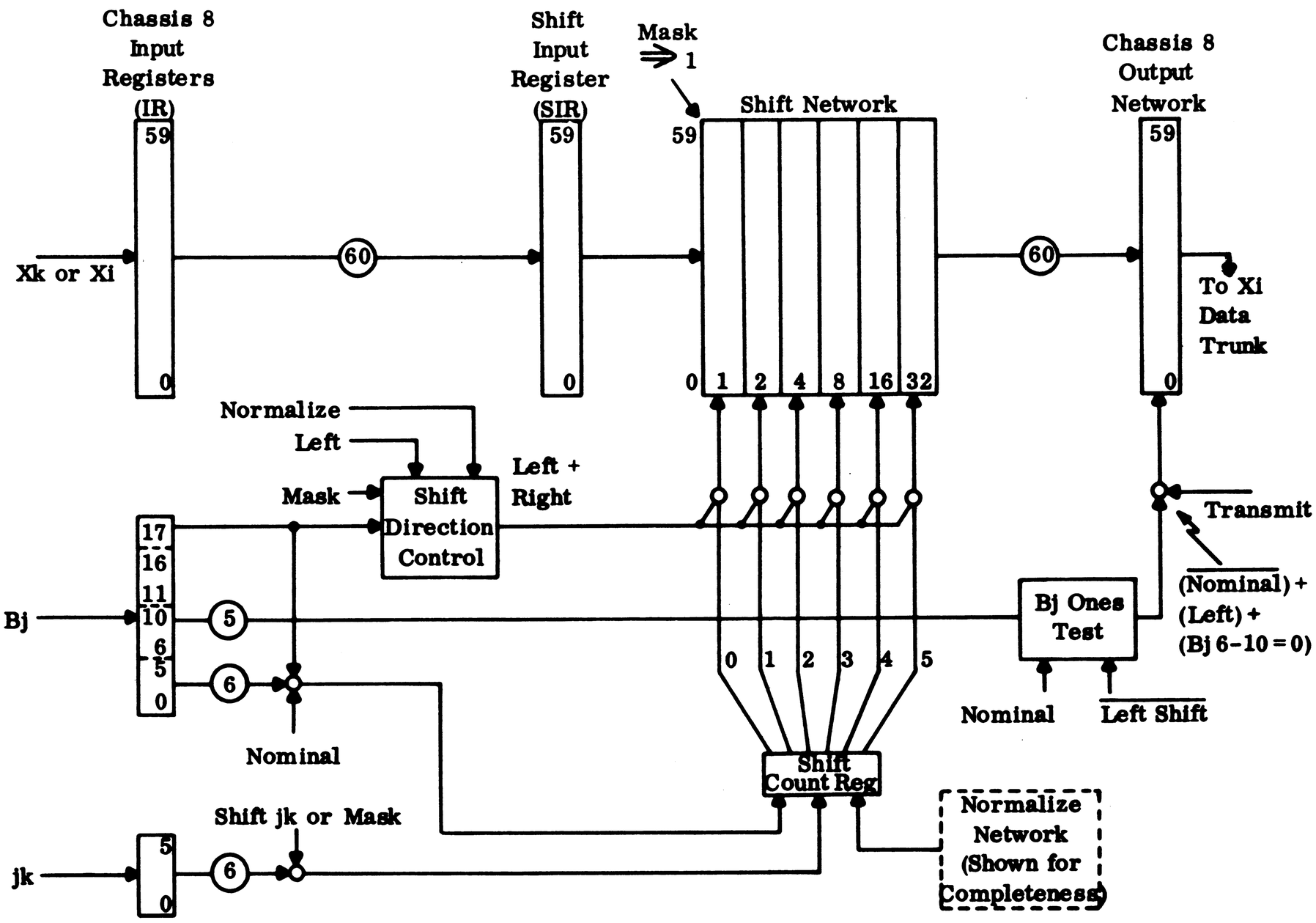


Figure 81. Shift Functional Unit Block Diagram 1
(for Instructions 20, 21, 22, 23, or 43)

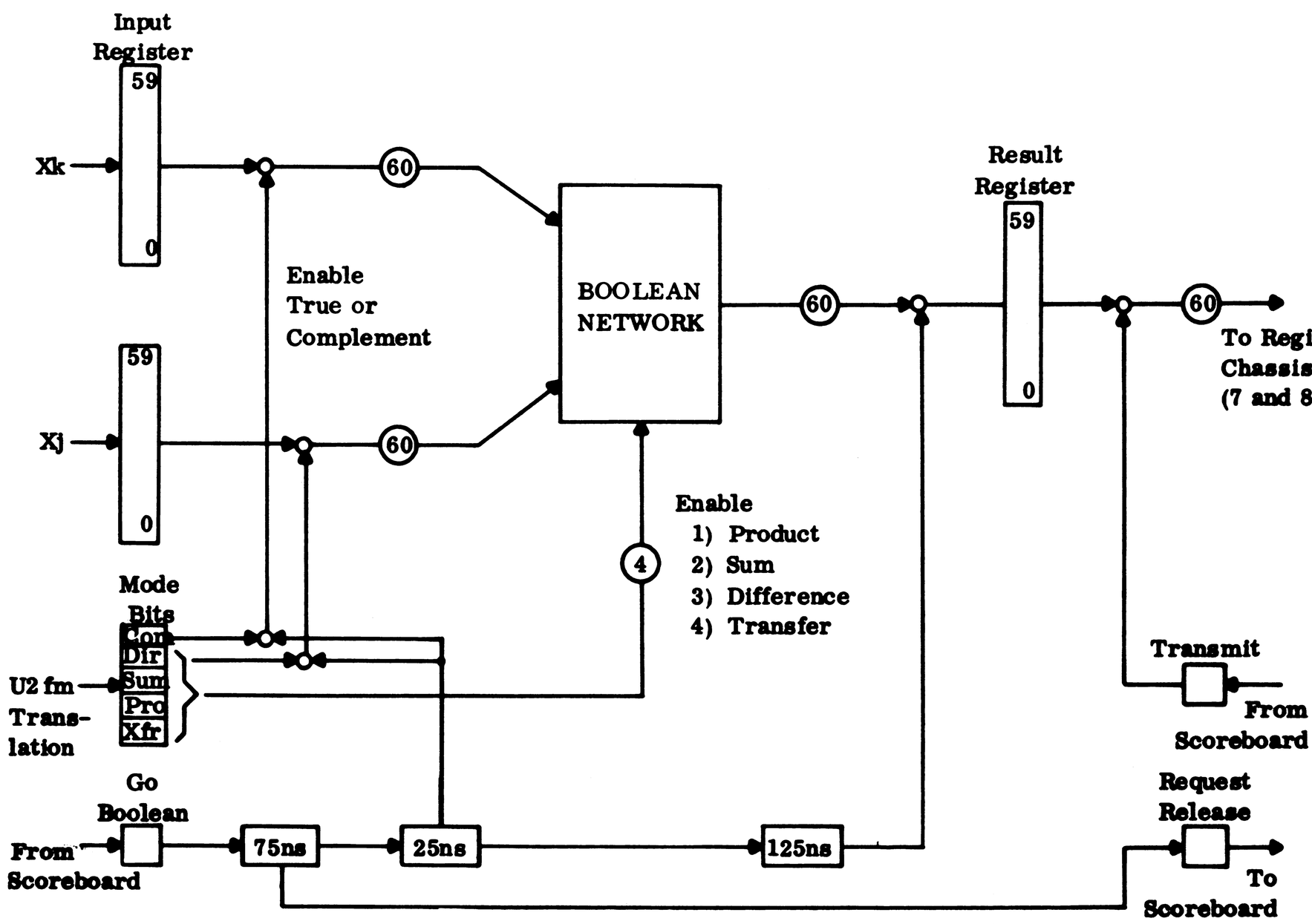


Figure 82. Boolean Functional Unit

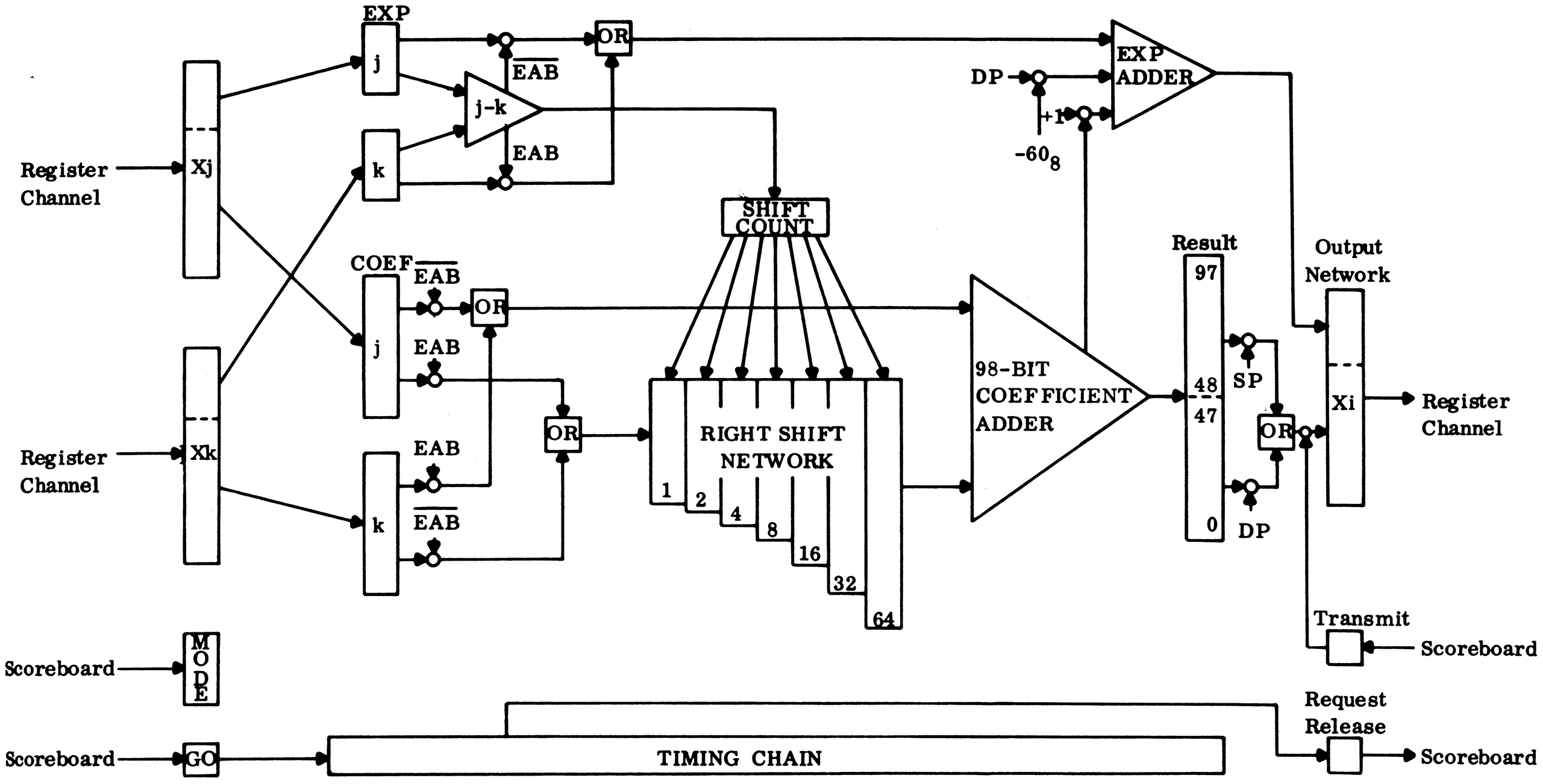


Figure 83. Add Functional Unit

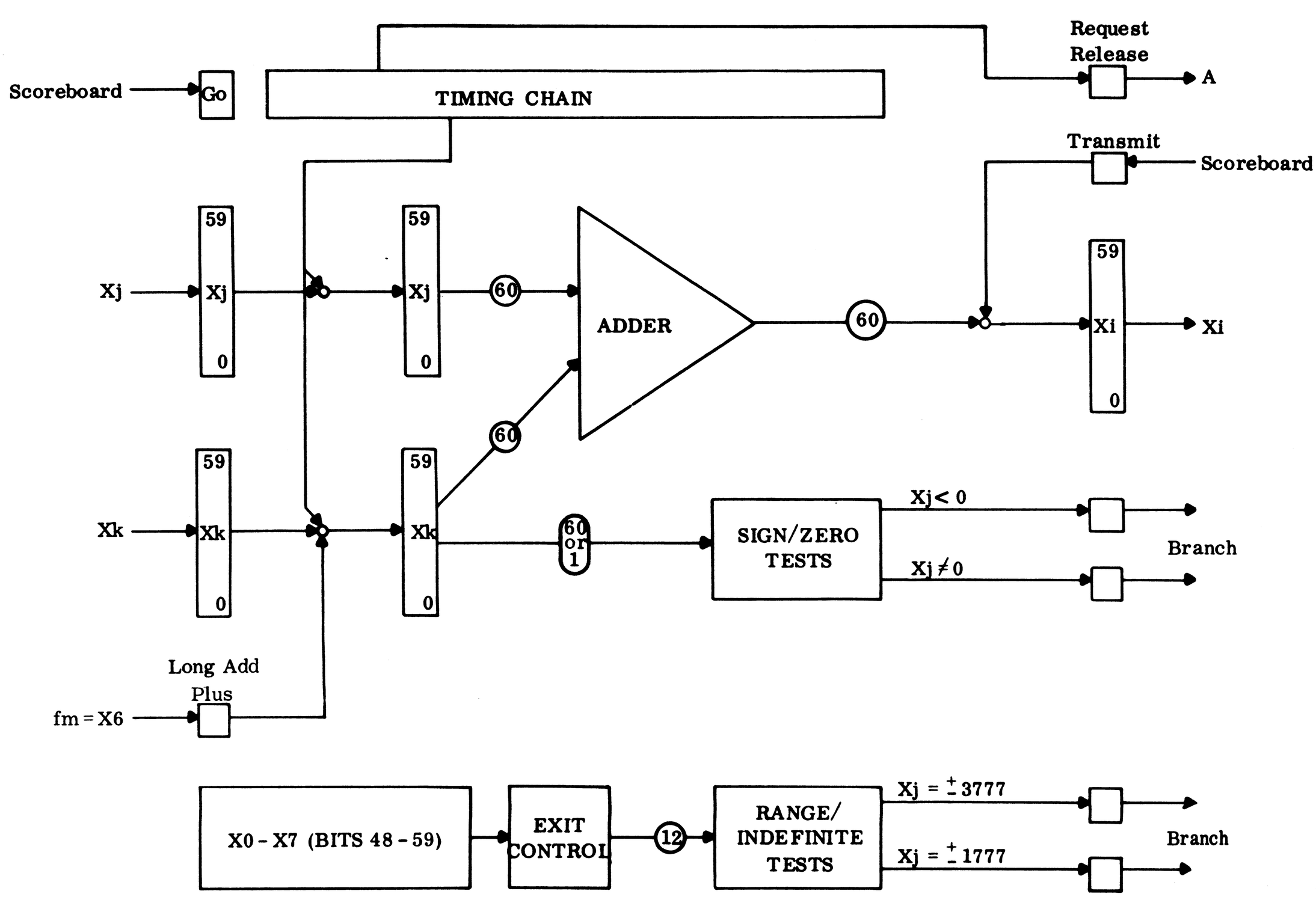


Figure 84. Long Add Unit Block Diagram

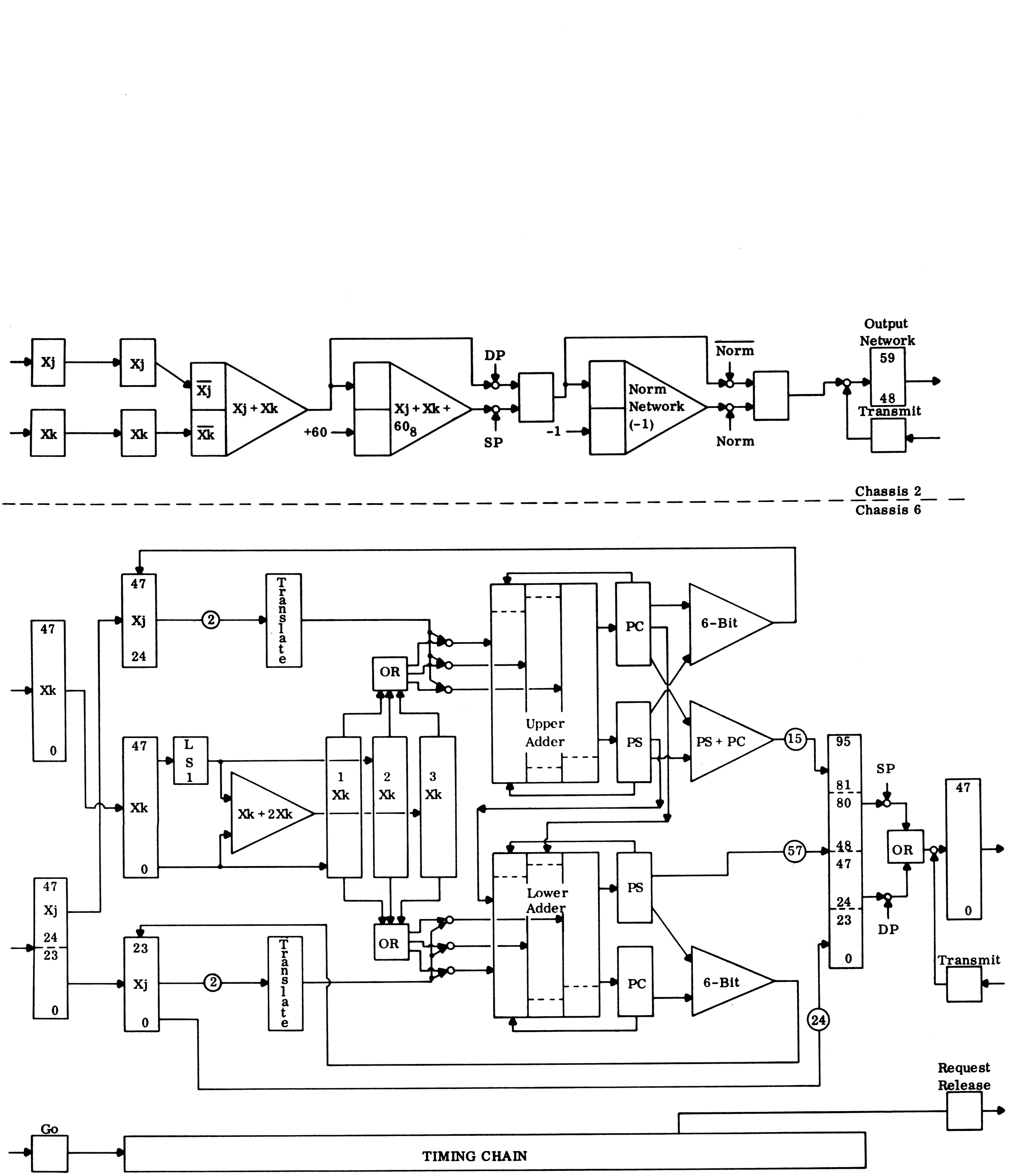


Figure 85. Multiply Functional Unit

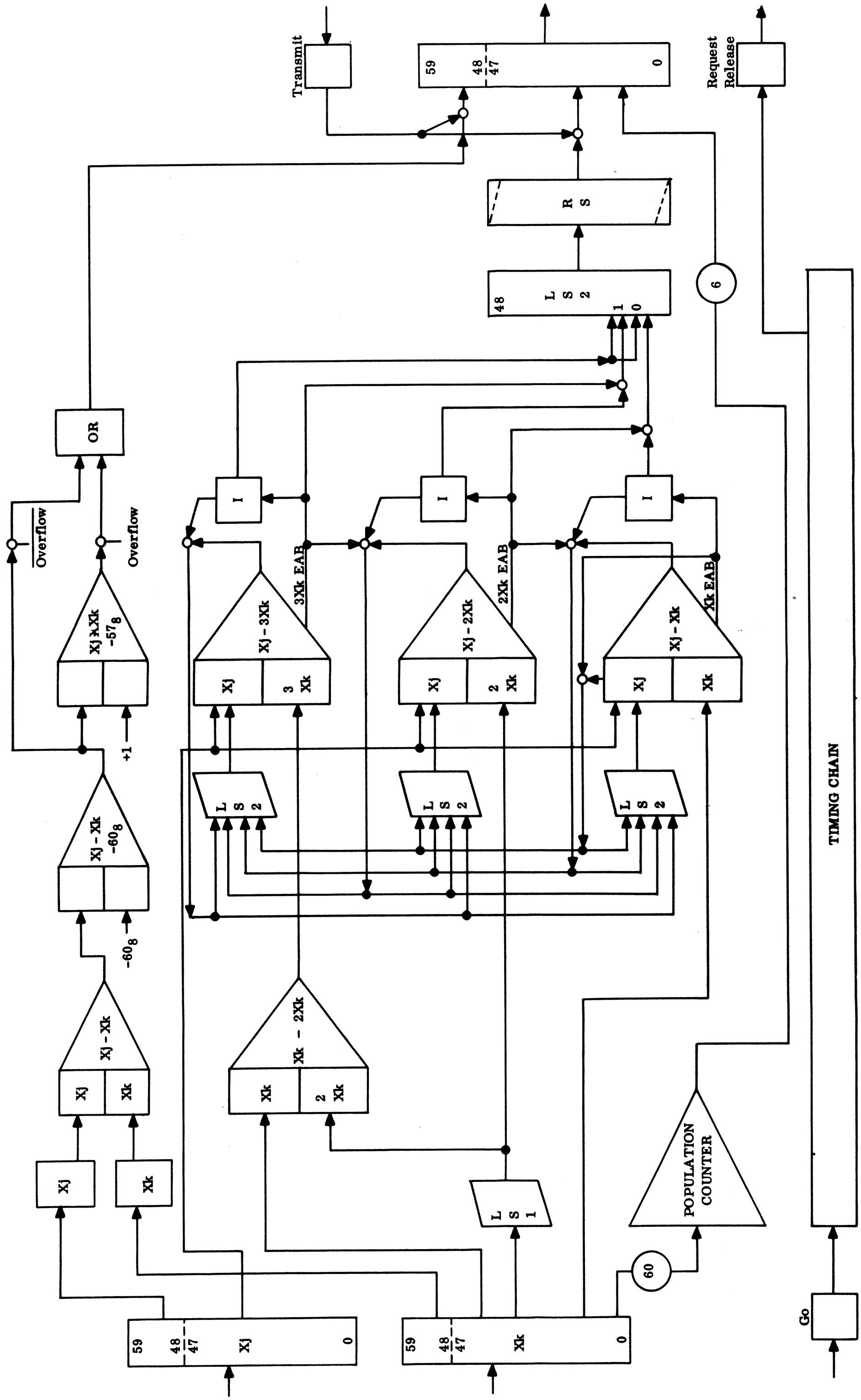


Figure 86. Divide Functional Unit

Figure 87. Increment Functional Units Block Diagram

Figure 88. Branch Functional Unit

CHAPTER 5

THE 6400 COMPUTER SYSTEM

INTRODUCTION

This chapter provides an introduction to the CONTROL DATA® 6400 Computer System.

MARKET POSITION

The 6400 is a large-scale, solid-state, automatic, parallel, digital computer aimed towards the general-purpose scientific market. Its input/output section is made up of 10 small general-purpose computers and 12 data channels which are accessable to any of these computers. These make the 6400 competitive in many areas.

6400 SYSTEM GENERAL BLOCK DIAGRAM

The general block diagram (figure 91) will help review the overall functional sections and their interconnections. In the basic 6400 system, the only way to place data and instructions into the central memory is through the peripheral processors (PPU's). The central processor (CPU) is controlled by the PPU's. When the CPU stops, only a PPU can restart it. The PPU's do not have a Stop instruction.

SYSTEM CHARACTERISTICS

The 6400 system characteristics may be divided into four parts: data channels, PPU's, central memory, and central processor.

® Registered trademark of Control Data Corporation

DATA CHANNELS

The 10 6400 peripheral processors are connected to 13 independent data channels. Twelve of these data channels are bidirectional and can be connected to I/O equipment. The other is permanently connected to the Real-Time clock and is used specifically for monitoring the clock. The Real-Time clock is a 12-bit, two's complement counter, advanced every one μsec.

The method used to connect the data channels to the peripheral processors allows any processor to use any channel. This scheme allows flexibility in handling I/O equipment.

PERIPHERAL AND CONTROL PROCESSORS (PPU's)

There are 10 identical peripheral and control processors (PPU's) associated with the basic 6400 Computer System. Each is independent of the other and each has its own 4K, 12-bit memory. The basic cycle time is one μsec. Each is capable of communicating with the central memory, the 12 I/O channels and the Real-Time clock. Each processor is capable of executing a 64-instruction repertoire. The instructions include logical, branch, I/O, central memory access, direct, indirect, and indexing addressing modes. Average instruction execution time is two usec.

CENTRAL MEMORY

The central memory has 60-bit words grouped into 4096 word banks, and it employs bank phasing. The three standard sizes available are 32,768 (8 banks); 65,576 words (16 banks); or 131,072 words (32 banks). The 60-bit word is obtained by arranging five PPU memory modules in parallel.

Therefore, the basic memory cycle time is the same as the PPU's; however, because of bank phasing, 60-bit words can be moved at a 100ns rate. The central circuits of central memory have a retry feature which allows retrying a bank that was previously busy every 300ns until it is free. The control circuit also has a scanner which allows sorting of simultaneous requests from different sources. Central memory can process a request every 100ns. Data can flow into or out of memory at a rate of 60 bits every 100ns.

CENTRAL PROCESSOR UNIT (CPU)

The 6400 CPU (figure 94) is a high-speed, binary arithmetic and control section. Types of instructions include: logical, branch, shift, increment, floating point, pack, unpack, normalize, floating point add, subtract, multiply, and divide (in single and double precision). All instructions and data come from central memory, and all new data (results of programs) is stored in central memory (the CPU has no I/O instruction). The instructions are executed in a sequential fashion. Instructions are either 15 bits or 30 bits in size, three-address type. The data word is always 60 bits. Data words are 59 bits and 1 sign bit in a fixed point format. Floating point has a 48-bit coefficient, 10-bit exponent, and two sign bits. Average instruction execution time is approximately one μsec per instruction.

SYSTEM ELEMENTS

The following list is a description of the various possible elements within a 6400 system.

- 6415A central computer includes 32,768 60-bit words of magnetic core storage, 10 peripheral and control processors each with 4096 12-bit words of magnetic core storage, and bidirectional data channels plus a data channel exclusively used for read time clock monitor.
- 6414A is the same as 6415A except it has 65,536 60-bit words of magnetic core storage.
- 6413A is the same as 6414A except it has 131,072 60-bit words of magnetic core storage.

SYSTEM BLOCK DIAGRAMS

There are several system configurations possible with a 6400 computer. Each configuration has certain advantages which depend on the needs of the user. In this section of the manual, discussions analyze the CPU theory of the operation but do not examine the logic to any detail. References are made to the logic in order to aid the reader in following the operation through the CPU. Following this discussion are some general comments concerning the CPU.

BASIC 6400 COMPUTER SYSTEM

The basic 6400 system includes a central processor, 10 peripheral processors and 12 data channels. This basic system can have three memory variations (32K, 65K or 131K) of 60-bit central memory.

SPECIAL PURPOSE CENTRAL PROCESSOR

A special purpose central processor can be added that improves the arithmetic capabilities of the basic system. This special purpose central processor can communicate with the standard PPU's and the standard CPU through the central memory. The special central processor shares the central memory on an equal basis with the standard CPU. If a second standard CPU is installed, a 6500 system is formed (figure 92).

6681 DATA CHANNEL CONVERTER

The 6681 allows the 6400 peripheral processors to communicate with 3000 Series I/O equipment. The converter acts as a signal interface and allows users who presently have 3000 Series computers or equipment to use them with the 6400 system. The 6681 must be the first equipment on the channel and can control up to eight pieces of equipment.

GENERAL TIMING COMMENTS

The 6400 system central processor has a unified arithmetic unit. Therefore, instructions in the 6400 central processor are executed in sequential fashion, with little concurrency. Factors influencing instruction execution time are summarized below.

- The next instruction word is read from central memory between the execution of the first and second instructions. This process takes two minor cycles to initiate (the remainder of the RNI is parallel in time with the execution of the second instruction).

RULE

Add two minor cycles for each instruction word in a program which does not have a jump.

- All execution times listed are complete. The times include getting the next instruction ready to execute. The jump and return jump times include reading up and preparing to execute the new instruction word.
- The Return Jump, Jumps, and Load/Store memory instructions pay a time penalty for being the second instruction of an instruction word. This penalty is caused by hardware limitations and is not due to memory bank conflicts.

Table 14. Second Instruction Word Penalty Time

Instruction	Second Instruction in Word-Time Penalty
Jumps taken	One minor cycle
Return Jump	Two minor cycles
Load/Store	Two minor cycles
Minimum execution of second instruction	Eight minor cycles

- If the second instruction references the same memory bank as (P+1), there is an additional penalty of three minor cycles due to bank conflict.
- A Store (not Load) as the first instruction of a word can cause a bank conflict with (P+1). If this occurs, the penalty is three minor cycles.

The rules for efficient coding in the 6400 are listed below.

- Put jumps in the upper parcel. This eliminates both the two minor cycle RNI penalty and the possibility of having a memory bank conflict with (P+1).
- Where possible, keep Load/Store instructions in the bottom two parcels.
- Loads and Stores in consecutive parcels do not cause memory conflicts with each other.

CENTRAL PROCESSOR (CPU)

The 6400 CPU (figure 94) is a very high-speed arithmetic device. It is capable of performing many arithmetic and logical operations and can communicate with the central memory and the peripheral processors.

UNIFIED CENTRAL PROCESSOR

Instructions are acquired from central memory, decoded, and executed in a serial manner. Since instructions are executed one after the other rather than in a parallel fashion using functional units, the CPU is considered to be a unified arithmetic device.

OPERATING REGISTERS

In order to facilitate the execution of each program step and to reduce memory access times, operands used during the execution of an instruction come from 24 flip-flop operating registers that are loaded and stored under program control.

The CPU's 24 operating registers are divided as follows:

- Eight 60-bit X registers that hold operands used for computation. The X registers have a direct access path from central memory.
- Eight 18-bit A registers. These registers are used for loading and storing operands in the X registers. These registers are known as the address registers. Changing an A register causes a memory reference for an associated X register. That is, changing A1 through A5 causes a memory read and X1 through X5 receive the operand. Changing A6 or A7 causes a memory write and the operand in X6 and X7

is stored. A0 and X0 are not considered in this memory scheme.

- Eight 18-bit B registers are used as index registers.

All of the previously-mentioned registers are unique to the CPU and are not associated with the PPU's in any way. Figure 94 illustrates the interconnections discussed.

CENTRAL PROCESSOR OVERVIEW

Figure 94 shows the CPU in a simplified manner. An explanation of each major part will aid in the understanding of CPU operations.

INPUT

All inputs to the CPU pass through this area. These inputs include instructions, operands, and control register contents.

INSTRUCTION CONTROLS

This area includes all instruction translation logic and the control sequences logic necessary to carry out the execution.

OPERATING AND CONTROL REGISTERS

The control registers include the Program Address register (P), the Reference Address register (RA), the Exit Mode register (EM), the Field Length register (FL) and the RAecs, FLecs, and MA registers for the ECS option. All of these registers are used in conjunction with the controls during the execution and progression of instructions.

The operating registers hold the operands used during the execution of instructions. There are 24 operating registers divided into three groups of eight each: the address registers (A), 18 bits in size; the index registers (B), 18 bits in size; and the operand registers (X), 60 bits in size. These registers are specified by the i, j, and k designators of each instruction and serve as the source and destination of operands for each executed instruction. This eliminates the need for a memory access to execute an instruction, and this greatly increases the speed of execution. The initialization of the operating registers is done by either the Exchange Jump or by execution of an instruction.

The X registers can also be loaded directly from memory and are also the focal points for operands that must be written into memory. This loading and storing of the X registers with memory makes use of the A registers. Five of the X registers, X1 through X5, are designated as being receivers of operands. The associated A registers, A1 through A5, serve as the address for the memory reference. The X6 and X7 registers can only store operands whereas A6 and A7 are the associated address registers. Whenever A1 through A7 changes because of an instruction designating these destination registers, a memory reference is automatically initiated. Whether a Load or a Store takes place depends upon which A register is changed.

LARGE ARITHMETIC SECTION (LAS)

The large arithmetic section is used during the execution of instructions using 60-bit operands, usually of floating point format. It includes all multiply, divide, logical, add, and shift kinds of instructions. The LAS includes two 108-bit registers, an adder, a shift network, a normalize network, a shift and iteration counter, and all the necessary select circuits.

SMALL ARITHMETIC SECTION (SAS)

The small arithmetic section handles instructions using 18-bit operands. These include increment instructions, jump instructions, and the exponent manipulation for floating point instructions. SAS includes two 18-bit registers, an adder, an address range tester, and all necessary select circuits.

OUTPUT

The output section is where all data and control must pass on their way to central memory and ECS. This section includes operands, operating and control registers for exchange jumps, all addresses developed by the P register or central addresses for operands, and extended core storage addresses.

CENTRAL PROCESSOR OPERATION

The following section of this manual explains the CPU. Each major area is described along with its operations.

INPUT

All inputs to the CPU must go through Catching Register 9 (CR9). CR9 is 60 bits in size and consists of ungated flip-flops receiving data from coaxial cable inputs. Each flip-flop has a three-way fan-out associated with it. Two of the outputs are gated; the third is not. Bits 48, 49, and 50 have an additional output for the Exit Mode register. CR9 is unconditionally cleared each minor cycle.

OUTPUT

All addresses and data pass through this area of the CPU. There are four basic sets of transmitters in this area. They are data transmitters (60), ECS address transmitters (24), read P transmitters (17), and central address transmitters (17).

I6 and I7 inverter groups are the fan-in circuits that collect all the necessary data that is sent by the data transmitters to central memory (usually on Exchange Jump). The read P transmitters send the contents of the CPU P register to the peripheral processors unconditionally every 100ns.

INSTRUCTION CONTROLS

The instruction controls section of the CPU receives, temporarily holds, decodes, and then executes instructions. The section contains the receiving, holding, op code translating, and sequencing logic.

All instructions flow from CM through the read distributor to CR9. From CR9 the instructions continue to the U1 register where they are disassembled. A 60-bit instruction word can be made up of four 15-bit instructions, two 15-bit and one 30-bit instructions, or two 30-bit instructions (figure 89).

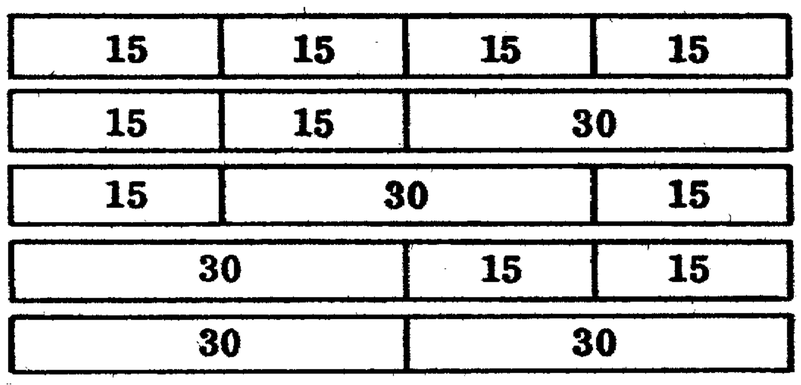


Figure 89. Instruction Position Variations

The disassembling of a 60-bit instruction word is called parceling. A parcel in the 6400 is a group of 15 bits. A 60-bit instruction word is grouped as shown in figure 90. Parcel 0 is always copied from U1 first, then parcel 1, parcel 2, and finally parcel 3.

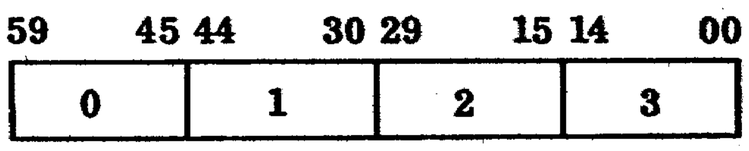


Figure 90. 60-bit Instruction Word Grouped in Parcels

A parcel counter circuit enables each parcel from U1 through U2 to the U2 translators and slaves. Normally, the enable is up for most of an instruction execution time. However, the gate is not made into U3 until the next parcel is needed.

One parcel is one 15-bit instruction. However, a 30-bit instruction needs two parcels. Because only one parcel is moved at a time, the upper 15 bits of a 30-bit instruction move to U3 first, then the parcel counter is advanced. This enables the next parcel, (the lower 15 bits of the 30-bit instruction) through U2. Notice that the U2 slaves also feed the I3 inverter group and that the lower three bits (k) of U3 also have a path to I3 k and the lower 15 bits make up the K of a 30-bit instruction. K is gated into I3 early during the execution of an instruction and the parcel counter is advanced again. Thus, the next 15-bit parcel is enabled through U2 and would be the next parcel gated into U3.

After parcel 0 and parcel 1 have been taken out of U1, the remaining 30 bits are sent to the RNI hold register. This register holds the last two parcels while the next 60-bit instruction word is read from CM. When the parcel counter equals two or three, the bits are enabled from the RNI hold register through U2. When the parcel counter is advanced from three, it will go to zero, enabling parcel 0 from U1. This will be the first parcel of the next 60-bit instruction word.

An RNI occurs after execution of the uppermost instruction in each 60-bit instruction word. If the uppermost instruction is 15 bits, RNI occurs between parcels 0 and 1. If the uppermost instruction is 30 bits, RNI occurs between parcels 1 and 2.

TYPES OF RNI OPERATION

Normally, obtaining an instruction for the 6400 CPU to execute requires a parcel from U1 or the RNI hold register contents to U3. This type of reading next instruction (RNI) operation is called parcel RNI.

After the first parcel RNI and the execution of the first instruction, the contents of the P register are advanced and added to the RA register contents to obtain the actual CM address of the next 60-bit instruction word. This addition needs the small arithmetic section circuits for only 200ns, after which time the CPU waits for CM. Reading up of the next 60-bit instruction word while the second (and possibly third and fourth) instructions are being executed is called a full RNI.

After jump instructions, (Exchange Jumps, Conditional, Unconditional, etc.) the next instruction is in the following 60-bit instruction word still in CM. Therefore, a parcel cannot be taken from U1 or RNI hold registers. The CPU must stop and wait until the next instruction is available in U1. This process of sending an address to CM and then waiting until it is in U1 is called an initial start RNI, or simply an initial start. Usually the address sent to CM would be (P) + (RA). However, the Return Jump instruction advances (P) during its initial start.

Unlike the parcel RNI, the initial start operation does not place an instruction into U3 and start executing it. Rather than duplicate a parcel RNI, the 6400 CPU stops an initial start after the word is in U1 and then starts a standard parcel RNI operation.

LARGE ARITHMETIC SECTION (LAS)

The LAS is used during execution of most central processor instructions. Most instructions using 18-bit

operands do not use this area of the CPU. The focal point of the LAS is the 108-bit D adder.

The C and D registers, each 108 bits in size, serve as feeder registers for the D adder as well as transfer paths used during instruction execution. The other important parts of the LAS are the I4 and I5 inverter groups which serve as fan-in, fan-out, and control inverters during instruction execution.

SMALL ARITHMETIC SECTION (SAS)

SAS is used for operations involving manipulation of 18-bit operands (increment instructions, jumps, exponent handling during floating point operations, and updating of P register during program execution). Central to this section is the 18-bit F adder.

The 18-bit E and F registers are the feeders. Other 18-bit registers (I0, I1, I2, and I3) serve as fan-ins from the various operating and control registers. I8, in addition to being a transient register for some operations of LAS, is used by SAS to forward addresses to central memory.

6400 CENTRAL MEMORY CONTROLS (CMC)

CMC is the overall control center of CM. It receives the address, requests, and other signals that may be necessary. It then initiates and controls the operation within itself, the CMB, and the data distributor until the operation is complete (figure 95). Figure 95 is a block diagram showing paths for addresses as they flow through CMC. The following paragraphs describe various elements of the CMC.

Catching Registers (CR0, CR1, CR2)

Catching registers are ungated flip-flops that receive data from a coaxial cable. Since the input is not gated, there is an ability to deskew data transmitted from chassis to chassis. The CMC has three such catching registers. They are Catching Register 0 (CR0) for central processor 0 addresses, Catching Register 1 (CR1) for central processor 1 addresses, and Catching Register 2 (CR2) for peripheral processor addresses.

CR0/1 receive all addresses sent by the CPU. The address may be for an operand (or an instruction) or may be the starting address for ECS transfer. CR2 receives all addresses sent by any of the PPU's. The address may be for Data Read (or Data Write) or may be the starting address for an Exchange Jump. CR0 and CR1 also are used as holding registers. For example, CR0 and CR1 are not cleared on normal CPU requests until after the address has been accepted. This is necessary since there may be an attempt to retry the address if no Accept is received. CR0 and CR1 also act as holding registers during an ECS transfer. During this operation, the updated new address formed by the address incrementor goes to CR0/CR1 as part of the recirculation path for the counter.

Peripheral Address Register (PAR)

The PAR is a secondary register associated with CR2. This register is necessary since the PPU's are unconditionally sending (A) to CMC every 100ns. Each word received by CR2 is automatically gated to PAR. When PPU's intend the word to be an address, they send a request along with the address. This request blocks the input gating term to PAR after the desired address is in PAR. The blocking input protects the address until it can be accepted by the CMB.

Gate Circuits

The gate circuits allow the movement of the addresses from a register to the proper point depending on the operation.

The memory address register gate (MAR gate) gates one of the four possible inputs to the Memory Address register. Such gating is used on all CMC operations.

The address incrementor register gate (AIR gate) gates one of the four possible inputs to the Address Incrementor register during Exchange Jumps and ECS operations.

The catching register gates (CR gates) gate the updated contents of the Exchange Counter register (EKR) from the exchange counter to the EKR, during Exchange Jumps.

The tag bit gate (TB gate) is used to gate a Go, No-Go condition to the tag bit sequences. This is used on all CMC operations.

Memory Address and Address Incrementor Registers (MAR and AIR)

Various other registers within CMC, such as the MAR and the AIR, are used as common registers during CMC operations. MAR receives all addresses that are to be transmitted to CMB. AIR serves as a holding register for AI. Such would be the case during Exchange Jump and ECS operations.

Address Incrementor (AI)

AI is an 18-bit two's complement counter. It is used to update the original address during an Exchange Jump and to update the ECS operations exchange counter (EK).

The exchange counter is used during Exchange Jumps only. It counts the addresses during an Exchange Jump to determine when the Exchange Jump is completed.

Tag Sequences

The tag sequences are used to identify memory references since several addresses may be operating in CMC Accept. The tag sequences are started when the address is transmitted to CMB. The sequence determines the time period before an Accept is expected. If no Accept is present, the retry circuit is initiated. There are several tag sequences. They are listed in table 15.

Table 15. Tag Sequences

Tag Bit	Description
Peripheral Tag Bit (PTB)	Present on PPU read or write operations.
Write Tag Bit (WTB)	Present any time there is a write operation intended from PPU's or CPU's (including Exchange Jump and ECS operation).
X Tag Bit (XTB)	Present during an Exchange Jump or an ECS transfer.
CPU 0 Tag Bit (C0T)	Present during CPU 0 request memory read and write and during peripheral exchange to CPU 0.
CPU 1 Tag Bit (C1T)	Present during CPU 1 request memory read and write and during peripheral exchange to CPU 1.

A tag sequence cycle time is 325ns, and it is initiated by the tag bit gate circuit. The Go to CMB is sent when the tag sequences are started. If the Accept has not returned before the near-end of the tag sequence, a retry will be initiated.

Retry

The retry sequence determines the time between Go signals for an address requesting CMB. If an address cannot be accepted because of a bank conflict, this address is resent to the CMB every 300ns until accepted. The retry sequence is initiated at the time the Go is sent to CMB. The retry sequence is made up of the retry 1, retry 2 and retry 3 FFs. Retry 3 sets only if there is no Accept received for the address that started the sequence. If retry 3 does set, the address and Go signal are sent to CMB. This also restarts the retry sequence.

Bank conflicts arise when an address is sent to a bank that is busy with a previous request. A bank is considered busy for the complete one μsec cycle time of that memory. The situation that causes the longest delay of a bank conflict occurs when four addresses try to access the same bank. The first address makes the bank busy and CMC receives an Accept within the first 100ns after issue. The second address issued to the bank has to retry three times before it can be accepted. The third address, issued 100ns later, retries six times before it can be accepted. The fourth address encounters the longest possible delay for an address because of bank conflict.

Accept Sequence

The accept sequence is initiated by an Accept signal returning from CMB. The Accept signals from all banks come into two accept 1 and accept 2 ungated FFs. The output of accept 1 and accept 2 are strobed into accept 3 every minor cycle. If the Accept is for a central processor address, the Accept signal is retransmitted back to the proper CPU determined by the tag sequence. This allows the CPU to communicate with the data distributor at the proper time. The output of accept 3 is sent to the retry circuit and inhibits the setting of retry 3. This stops the sending of successive Go signals for that address. The output of accept 3 is also sent to a circuit that examines it with respect to the peripheral tag bit sequence for the purpose of sending back the Resumes to the PPU's. The output of this circuit also starts the PPU read or write sequences that communicate with the CMB data distributor which times the placing or removing of the data in the distributor for PPU operations. The remaining flip-flops in the accept sequence (accept 4 through accept 12) are used for the All Quiet network. The All Quiet is used to determine the starting of an Exchange Jump.

Scanner

Within CMC, some consideration must be given to the possibility of the PPU's, CPU 0, CPU 1, and ECS making simultaneous requests. These simultaneous requests must be recognized and sorted out to utilize the central memory in the most efficient manner possible. This sorting operation is done by the scanner.

The scanner contains two flip-flops interconnected to form a closed loop. At first glance, with no requests, the circuit appears to be free-running. The first flip-flop sets the second, the second clears the first which then clears the second, which then resets the first flip-flop, etc. However, in actual operation, the scanner locks up in a position with the set and cleared sides of both flip-flops equal to a one output. This method is faster than a free-running scanner. Assuming no requests are present, and a request occurs, the time needed for it to be recognized and enabled to leave the module is less than 20ns.

Central memory control recognizes requests from either the PPU's, CPU 0, CPU 1, or ECS and generates a Stop Scanner signal corresponding to the device

which made the request. Assuming a PPU request is present, the scanner stops with both bits of the counter cleared (00_2). This gives a counter translation of Scanner = PPU, which allows the PPU request to actually start a CMB reference. At this time, all other requests have to wait until the scanner is released. The next device to have its request honored is CPU 0, then ECS, and finally CPU 1. Normally, the scanner only remains locked up for about 75ns. However, when an Exchange Jump is being executed, the scanner is stopped until the exchange is completed (approximately 1.6 μsec). This ensures the rapid completion of the Exchange Jump by eliminating memory conflicts.

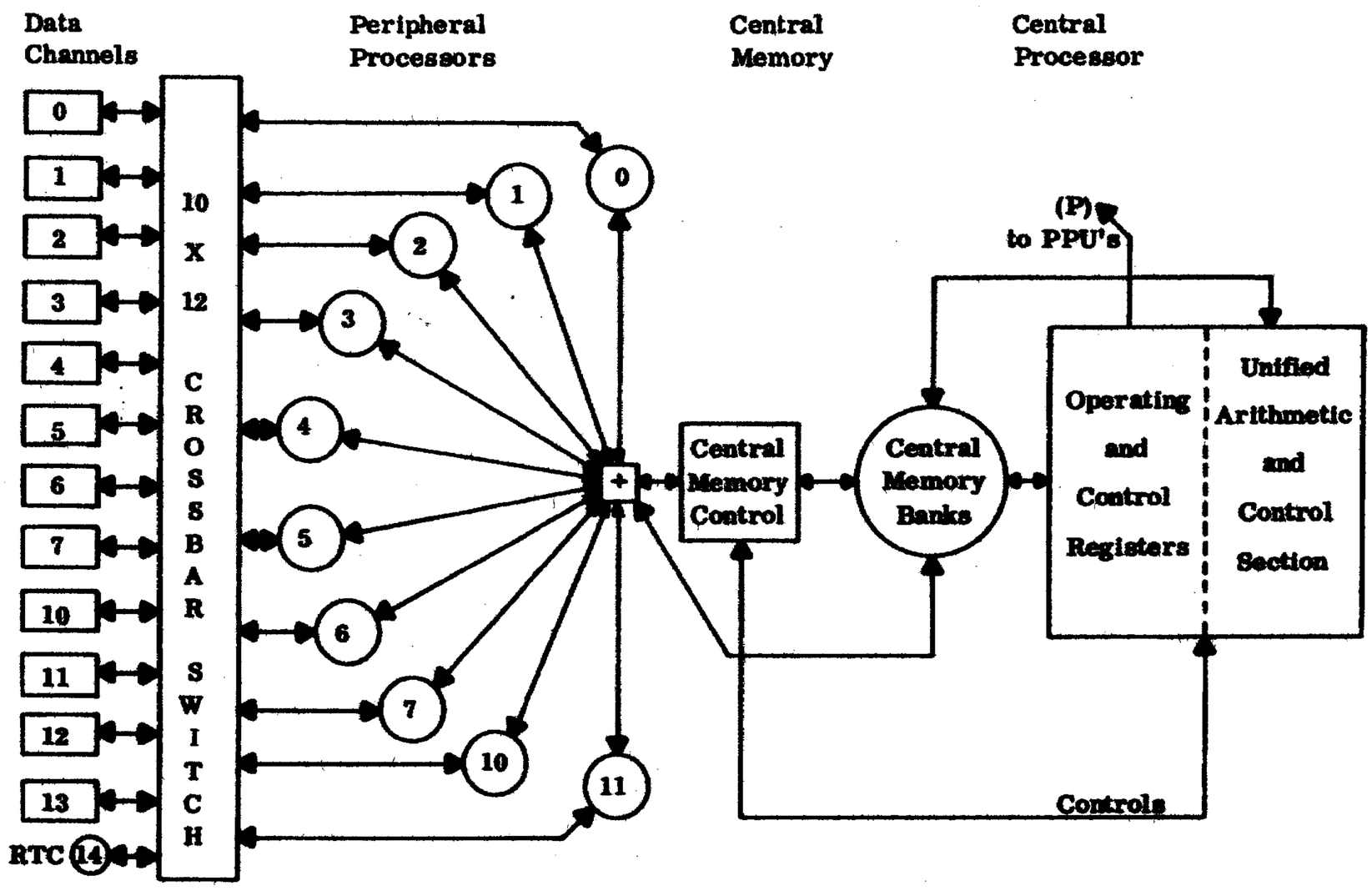


Figure 91. 6400 System General Block Diagram

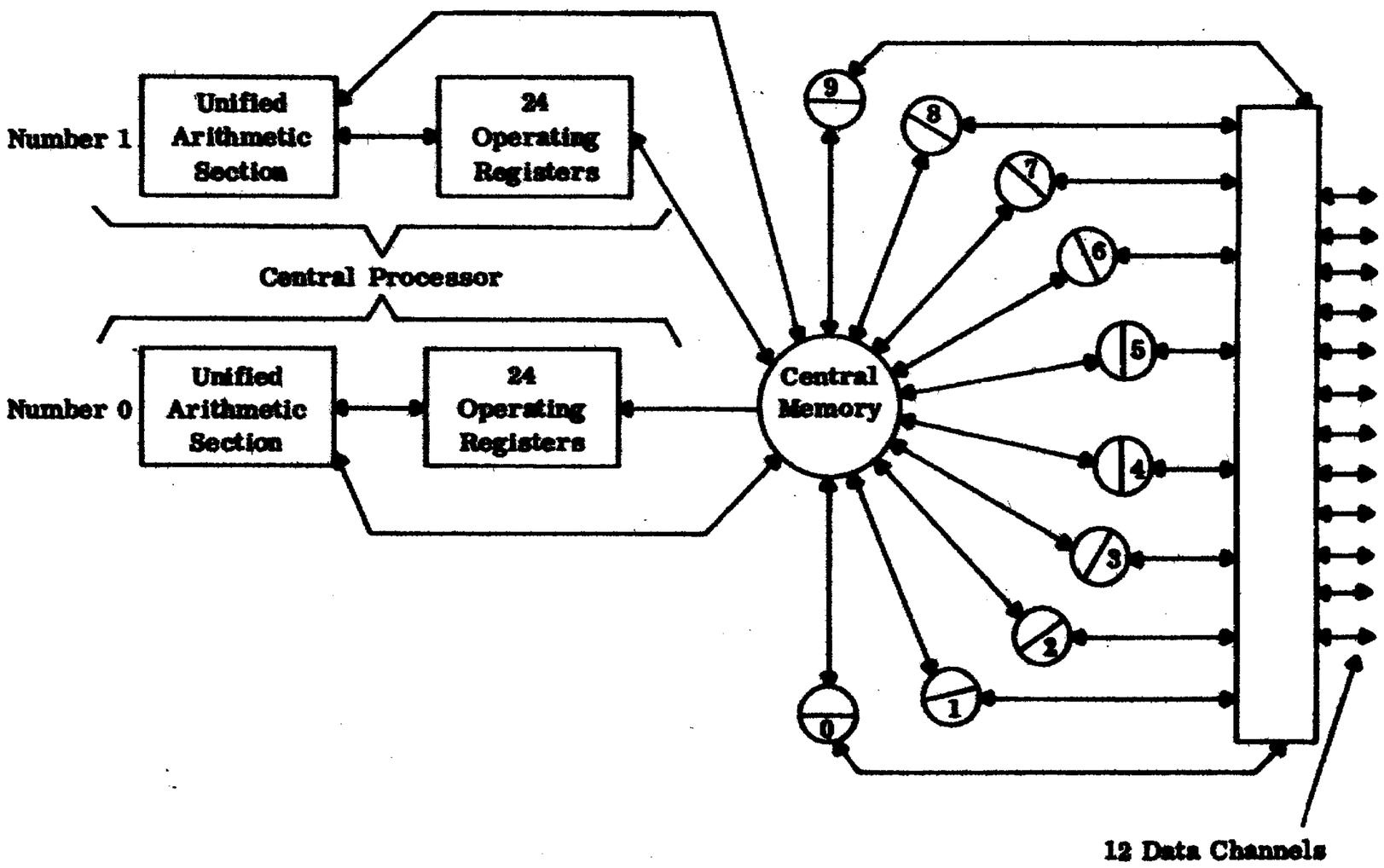


Figure 92. 6500 System General Block Diagram

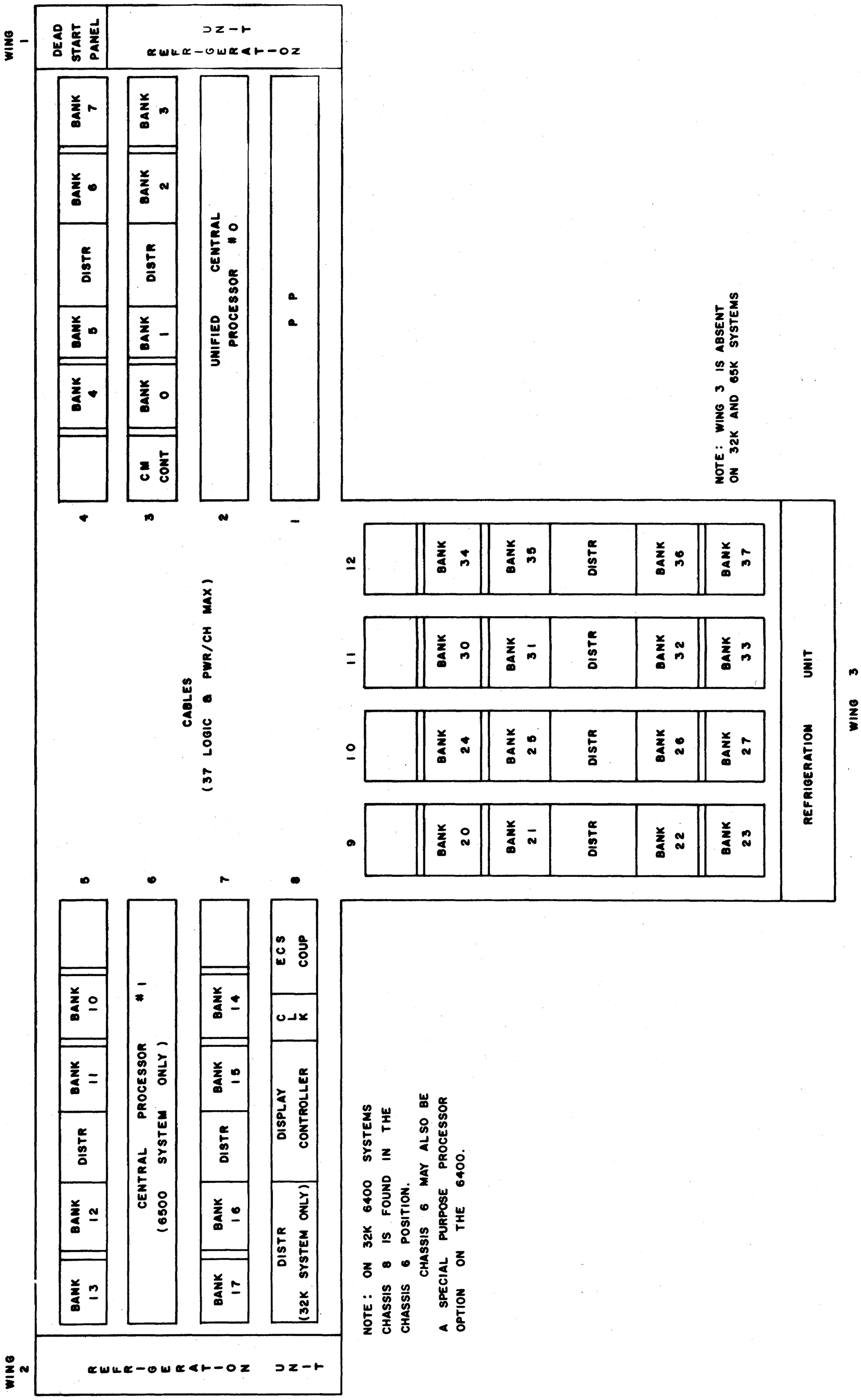


Figure 93. 6400/6500 Chassis Layout (131K)

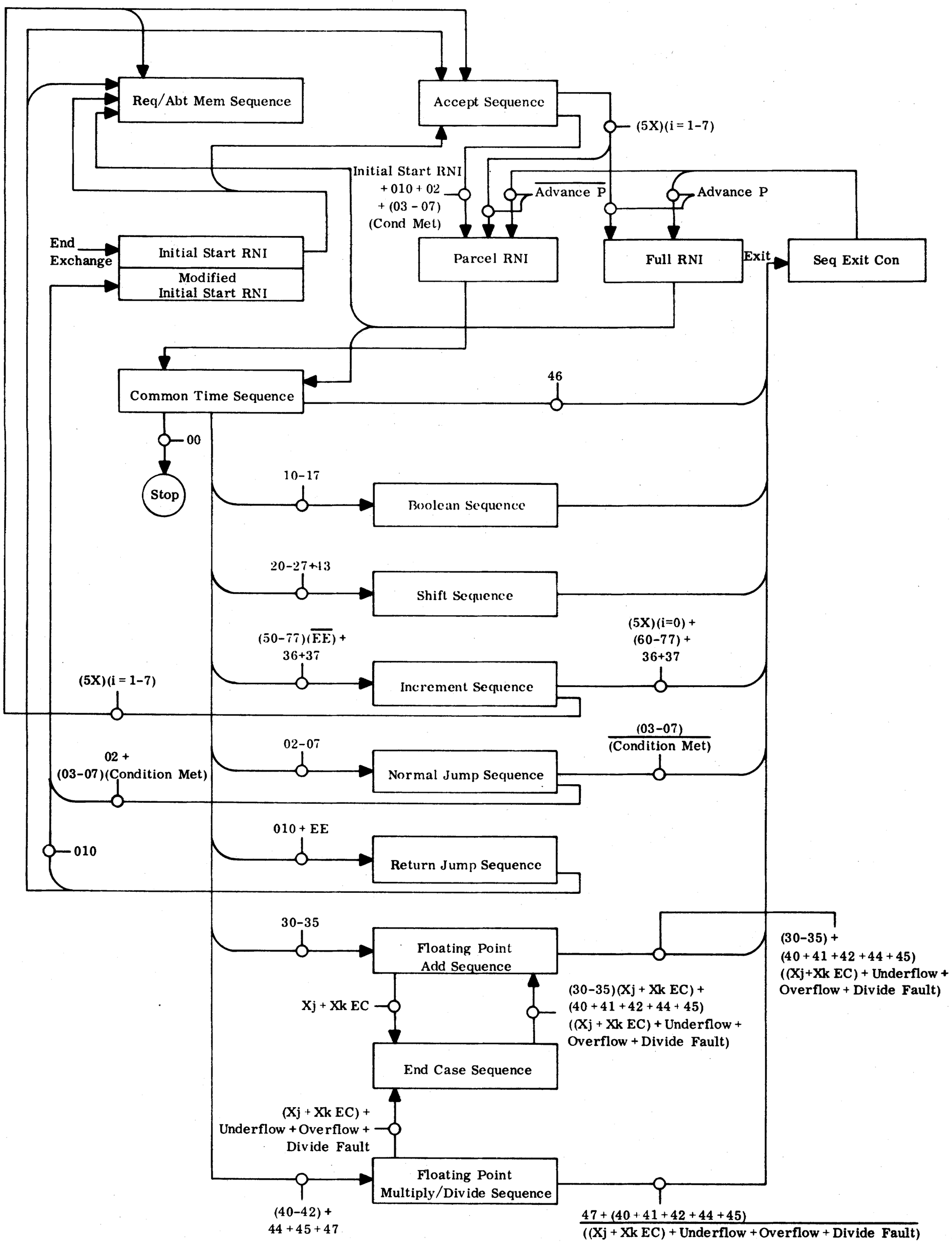


Figure 93a. 6400 CPU Sequence Control Diagram

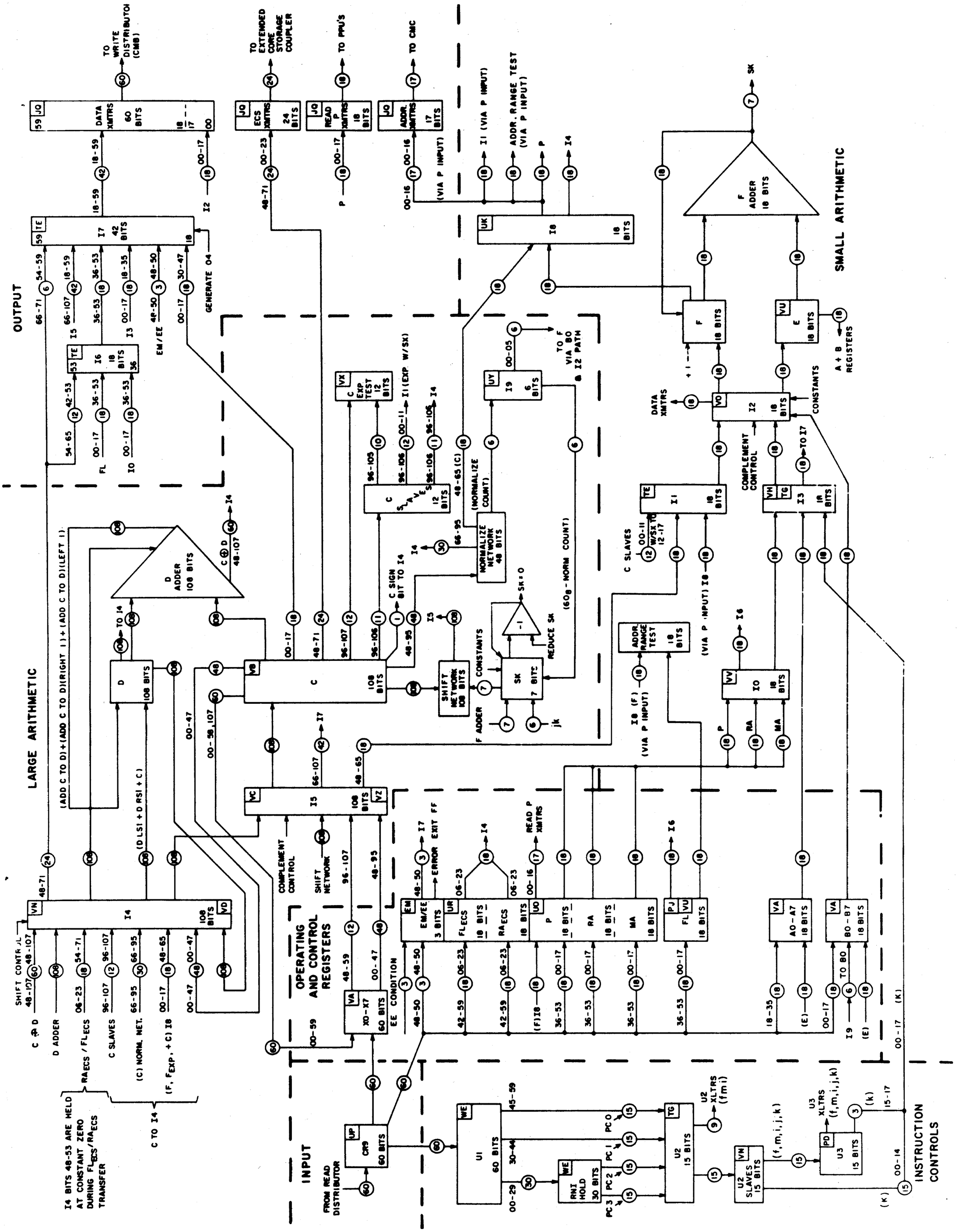


Figure 94. 6400 Central Processor Block Diagram

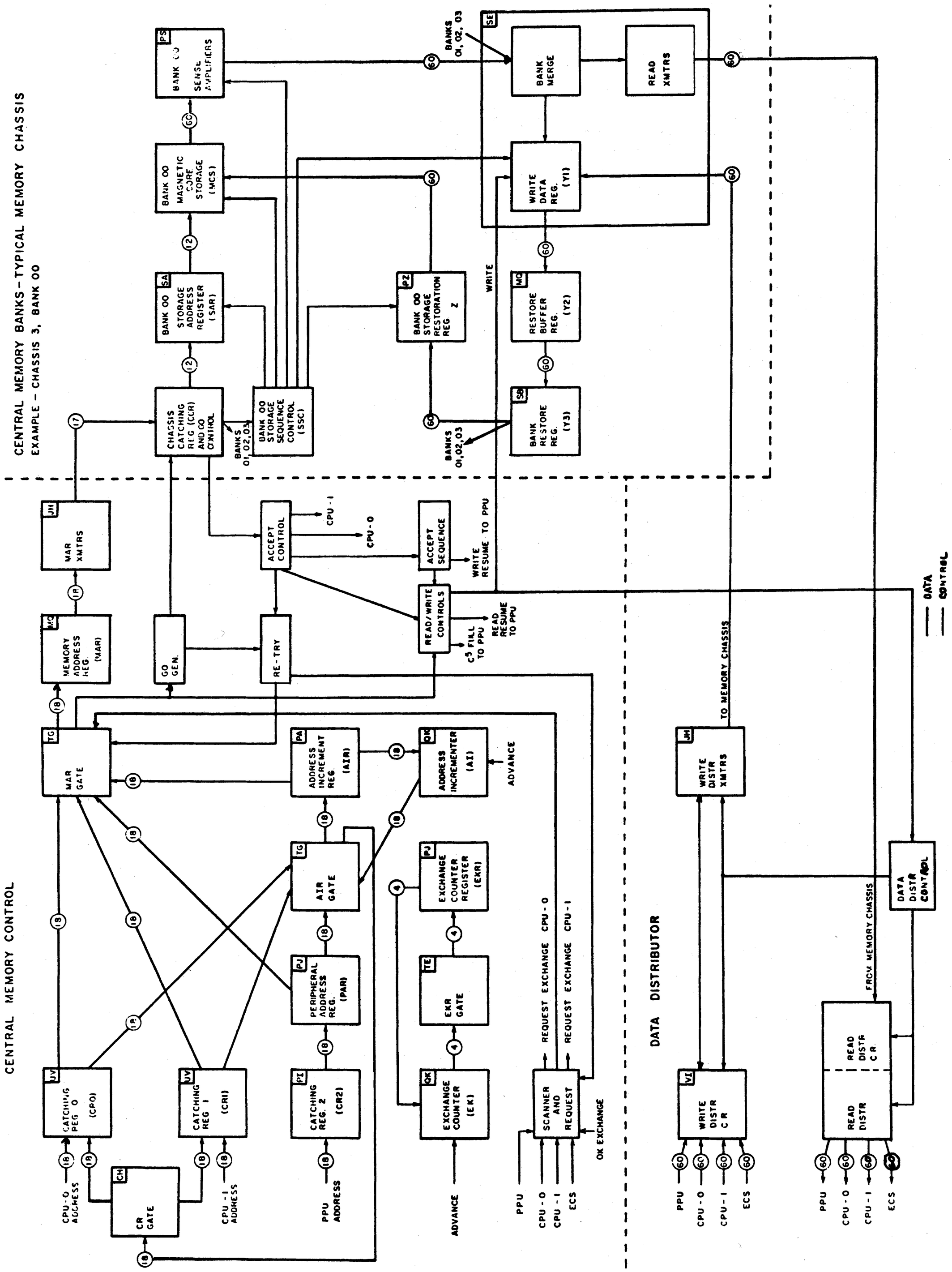


Figure 95. Central Memory Block Diagram

CHAPTER 6

6000 SERIES MAGNETIC CORE STORAGE ANALYSIS

INTRODUCTION

This chapter describes the magnetic core storage (MCS) areas which are used in the 6400 Computer System. The three general areas are peripheral processor memory, central memory (CM), and extended core storage (ECS).

PERIPHERAL PROCESSOR MEMORY

Each peripheral processor has its own independent 4K 12-bit word memory. One memory module makes up each memory in the PPU's. The complete cycle of any PPU memory is one μsec.

CENTRAL MEMORY (CM)

The central processor uses the central memory which can vary in size from 32K to 131K 60-bit words. Five memory modules logically ganged together make one 4K (60-bit word) bank of central memory. The complete memory cycle for any bank is one μsec. Since the central memory is bank-organized, greater speed in retrieving data is possible. This is accomplished by overlapping bank accesses. Speeds of 100ns per data word are possible by accessing different banks in sequence.

Peripheral processors are also able to access the central memory. Communications between the CPU and the PPU's are accomplished through this multiple access scheme of the central memory.

EXTENDED CORE STORAGE (ECS)

ECS is an optional item on the 6000 computer systems. It is intended as a fast method of back-up storage for central memory. The extended core storage can vary in size. However, there are some basic facts that can be considered.

The memory cycle time of the ECS is 3.2 μsec per 488-bit word. The 488-bit word is equivalent to eight central processor words, and each has its own parity bit. By using this large word size, the slower cycle time is not considered since memory data flows between ECS and central memory at 100ns rates. The ECS is ideally suited to the 6000 computer systems since the data trunks that carry data between the central processor and central memory are taxed very little by the central processor.

Elements of the ECS include the ECS banks, the ECS multiple-access controller, and the coupler associated with the central memory.

DETAILED DISCUSSION OF CORE STORAGE SYSTEMS

Study of the magnetic core storage systems of the 6000 computer systems involves examination of several areas of the logic and electronics. The areas to be considered are listed below and explained in detail on the following pages.

- Logic and electronics of the storage module
- PPU memory scheme and control
- Central memory control (CMC)
- Central memory banks (CMB)
- Central memory read and write distributors
- ECS controls
- ECS coupler
- ECS controller
- ECS banks

STORAGE MODULES

The storage module stores 4096 12-bit words. The module uses magnetic cores and coincident current techniques in reading and storing data in the cores. The complete time (cycle time) to read or store one word is 1000ns.

MAGNETIC CORE MEMORY

GENERAL PRINCIPLES

The memory uses the permanent magnetic properties of ferrite cores to store the bits of computer words. A magnetic core is a bistable device capable of storing a one or a zero, depending upon its state of remanent magnetization.

Wires carrying current pass through the cores and magnetize them in one direction or the other. The direction of current flow determines the direction of magnetization. Approximately 400 ma turns of magnetizing force are required to switch a core.

The cores are assembled in a square matrix or memory plane. Five wires pass through each core. They are a horizontal drive (H) wire, a vertical drive (V) wire, a horizontal inhibit (I) wire, a vertical inhibit (I) wire, and a diagonal sense (S) wire. The coincident current switching technique is used. A core is addressed by simultaneously driving half-amplitude (200 ma) current pulses through a selected H wire and a selected V wire.

Only one core among all those in the matrix is subjected to a magnetizing force sufficiently large to switch its magnetic state. This core is at the intersection of the selected H and V wires. All other cores in the same row or column as the selected core receive half-amplitude current pulses and are said to be half-selected. In figure 96, if the left-most V wire and the top H wire carry current pulses, core A is fully selected, cores B and C are half-selected, and core D is unselected.

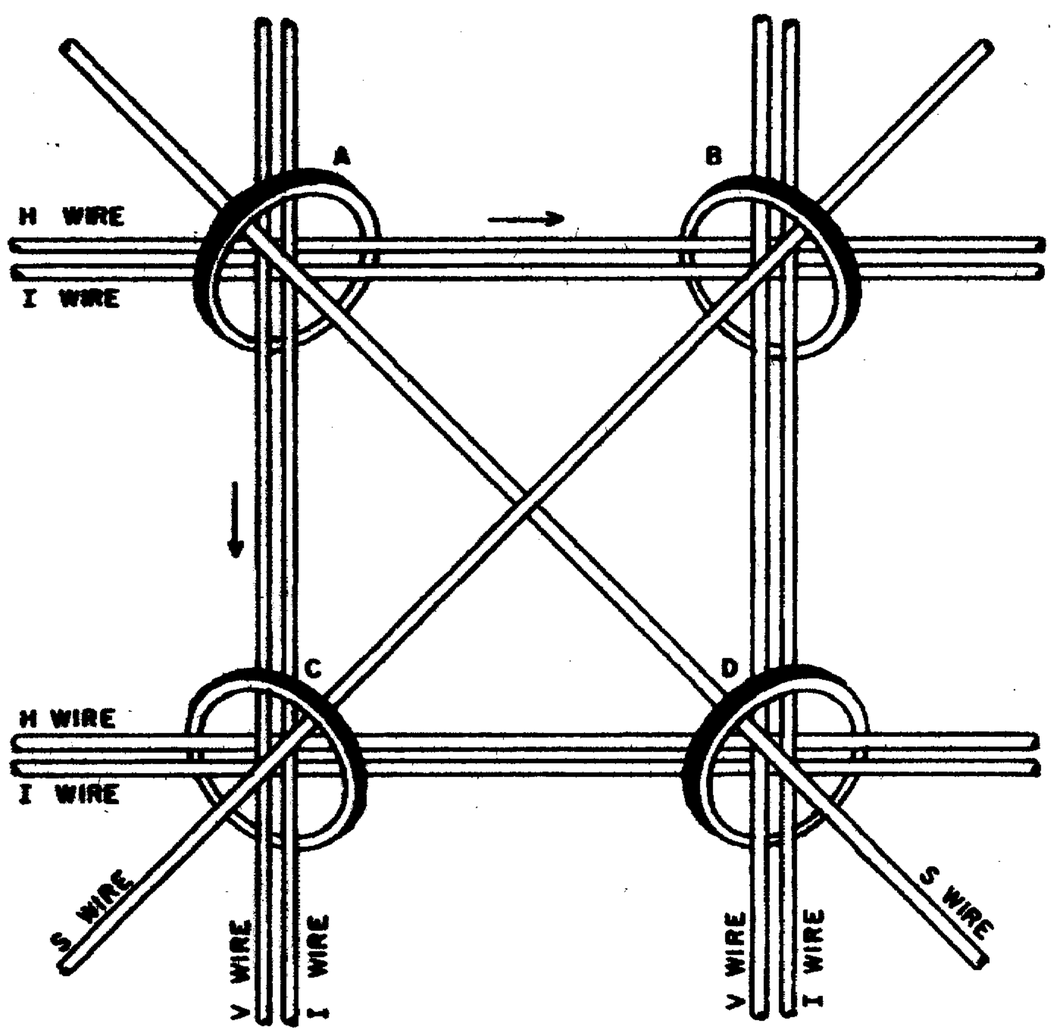


Figure 96. Core Selection

The polarity of residual magnatization determines the binary information stored in a core. The magnetizing force of a read current pulse on the selected H and V wires stores a zero. The absence of a pulse on the I wires permits a one to be stored by the write pulse.

Applying a read pulse to the selected H and V wires extracts or reads information from the selected core. If the core previously stored a one, the pulse drives it to zero. This change induces a pulse in the S wire. The pulse in the S wire is interpreted as a one bit from the core. If the core previously stored a zero, the read pulse does not affect it and no pulse is induced in the S wire. This absence of a pulse is interpreted as a zero bit from the core.

One memory plane holds 4096 cores in the 64 by 64 array. To use the coincident-current technique, each bit of a word is stored in a separate plane. Therefore, 12 bits require 12 planes.

DESCRIPTION OF MAGNETIC CORES

The magnetic properties of a core are represented by its hysteresis loop (figure 97) which plots magnetic flux density (B) as a function of the field intensity (H).

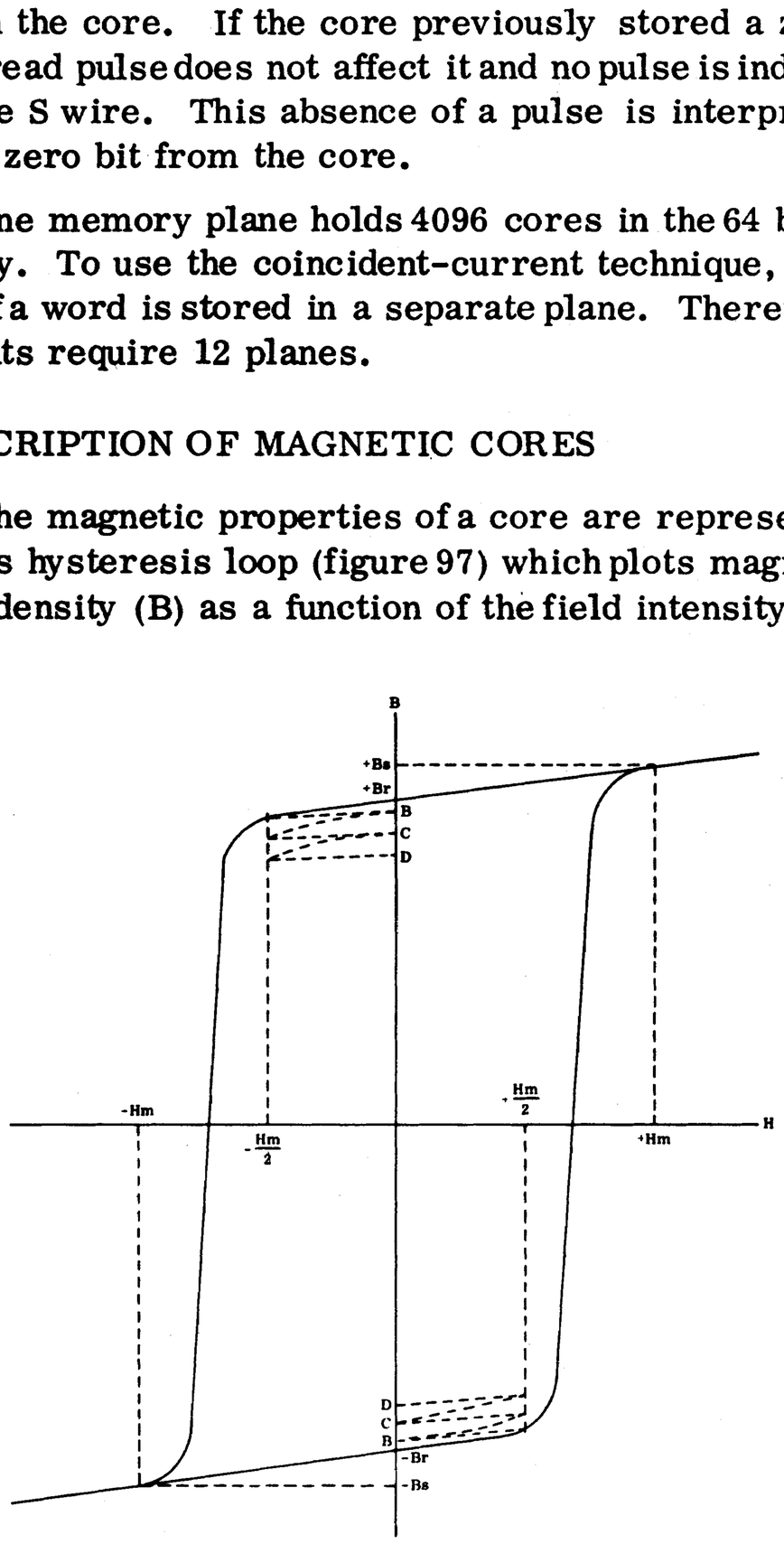


Figure 97. Hysteresis Loop

If current flow sufficient to cause a field intensity of +Hm is applied to the drive lines, the flux density increases to saturation (+Bs). When the current is removed, the flux density drops to the residual positive value (+Br), which is the zero state, and remains there. Another pulse of +Hm merely shifts the core to +Bs again. After the pulse is removed, it drops back to +Br. Application of current flow sufficient to cause a field intensity of -Hm reverses the flux density to -Bs. When the current is removed the flux density drops to the residual negative value (-Br), the one state.

The basic memory cycle is composed of half-amplitude pulses. Each pulse is capable of producing a field intensity of Hm/2. A half-amplitude pulse is insufficient to switch the core because the flux density returns to the residual value, or a slightly lower value, after the pulse is removed. The coincidence of two half-amplitude pulses, one on the H drive line and the other on the V drive line of a core, produces a net field of Hm which is sufficient to switch the core. When a half-amplitude pulse drives the flux density toward the knee of the hysteresis loop, the flux travels up (or down) the knee somewhat and then returns to a slightly lower residual value such as B. Since the core is now operating on a smaller loop, further half-pulses reduce this remanent flux again. However, this effect soon reaches a limit, as at point D. Core characteristics are such that the hysteresis loop is nearly square and half-amplitude pulses produce a small flux change. The smaller the flux change, the smaller the induced voltage on the sense line. This lowers the noise on the sense line due to unselected cores.

Any change in the magnetic state of a core causes a change in the total flux linking the core and any winding passing through it. Such a change produces a voltage output on the sense winding (figure 98). During the period that H is applied, the voltage is sampled to see if the core switches. If a large voltage is sensed, the core was in the one state and has switched. If only a small voltage is sensed, the core was in the zero state and has merely shifted from +Br to +Bs and back again.

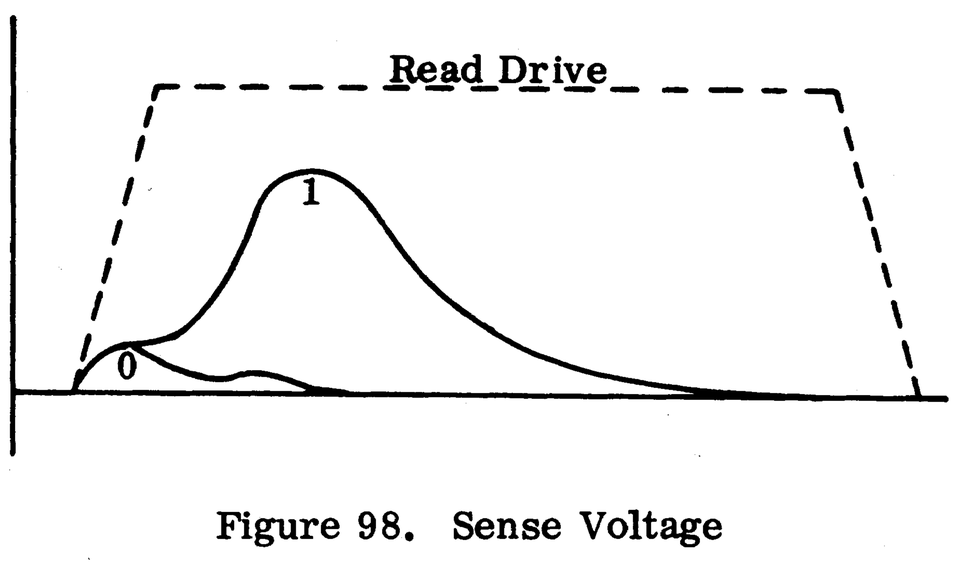


Figure 98. Sense Voltage

All 4096 cores in a plane are strung on a common sense line; 64 horizontal cores and 64 vertical cores contribute to the sense line voltage when reading. The total voltage induced by the 126 partially selected cores is less than the voltage induced on the line by the one core which has switched. The sense line is strung diagonally through the cores of a plane in such a way that voltages from unselected cores tend to cancel.

STORAGE MODULE CONSTRUCTION

A storage module consists of several assemblies connected mechanically and electrically. These assemblies are listed below.

- Memory plane (total 12) each of which holds 4096 magnetic cores
- Stack which is the 12 memory planes assembled
- Drive deck (2) which contains circuits which supply current to the horizontal and vertical drive lines
- Inhibit deck (4) which contains circuits which supply current to the horizontal and vertical inhibit lines
- Connectors (2) to provide electrical access to storage module circuits
- Miscellaneous mechanical parts for assembly and cooling.

Registers external to the storage module hold the 12-bit address and 12-bit word to be stored. External (to module) amplifiers terminate the output of the 12 double-ended sense lines and distribute the 12-bit read word.

MEMORY PLANE

A memory plane is a 4 by 4 inch printed circuit board which holds 4096 cores in a 64 by 64 matrix. The cores are strung in an alternating pattern, and five wires pass through each core. The wires terminate on the inside edges of the board. Printed wiring carries the connections to the outside edges (top and bottom) for easier access.

The 64 horizontal drive line connections are brought to connections on each of two opposite sides of the board. The 64 vertical drive lines are similarly arranged on the other two edges. The sense line is a single wire threading all cores, and its two ends are brought out at one corner of the board. Inhibit line connections are brought out to the other three corners.

STACK

The 12 memory planes are assembled into a stack about two inches thick. A thin metal plate between each plane and on the front and back of the stack provides shielding and acts as a cold plate for cooling.

The drive line connections are made as shown in figure 99. Each drive line threads all 12 planes and returns to its drive deck. Horizontal and vertical drive decks are identical and are located at opposite ends of the stack.

The vertical drive deck mounts on the front of the stack and the horizontal drive deck mounts at the back of the stack and is rotated 90 degrees (CCW). The assembled stack allows extremely short drive (and also inhibit) lines and hence a low inductance. The latter is a contributing factor in the cycle time or rate at which memory can be referenced and respond.

The sense and inhibit lines are distributed evenly at the four corners of the stack for easy connection to the drive and inhibit decks and connectors. The even distribution results from rotating a plane 90 degrees with respect to an adjacent plane.

DRIVE DECK

One drive deck supplies current to the horizontal drive lines and the other deck supplies current to the vertical drive lines. The two decks are identical.

A deck has two printed circuit boards with printed wiring on both surfaces of a board. Components are mounted on and between the boards. A deck contains a read and a write current source and circuits to select each of the 64 drive lines.

INHIBIT DECK

The four inhibit decks supply current to the horizontal and vertical inhibit lines. The inhibit decks are of two types and the types differ only in printed wiring layout.

Inhibit lines on a plane are arranged in four 16-line groups in horizontal and vertical directions. Each deck has six current sources and circuits to select one of four groups on six planes.

CONNECTORS

The two connectors provide electrical access to the storage module.

MECHANICAL PARTS

Mechanical parts join the several electrical assemblies. The face plate supports the rest of the module and connects to the cold bars in the chassis. Various metal plates within the module also become cold bars through conduction. Component heat in the module flows by conduction and convection to the chassis cooling system. The connectors are mounted on the rear plate and secure to mating connectors in the chassis.

ADDRESS SELECTION

A memory reference provides a 12-bit address and read or write control information (i.e., whether the address is to be read or stored). All memory references cause a complete storage cycle which consists of read drive, inhibit drive, and write drive. Figure 100 shows memory cycle timing.

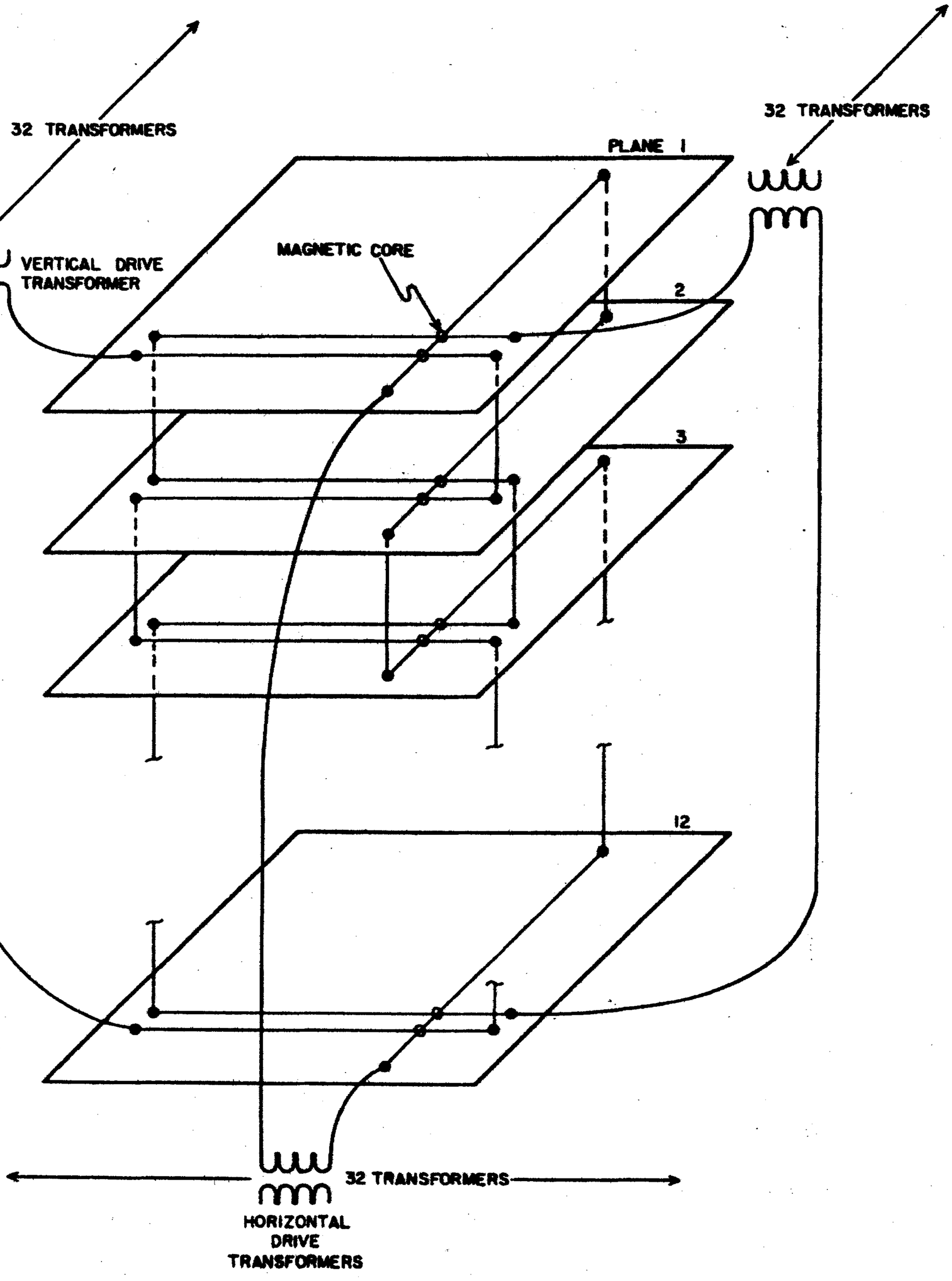


Figure 99. Drive Line Connections

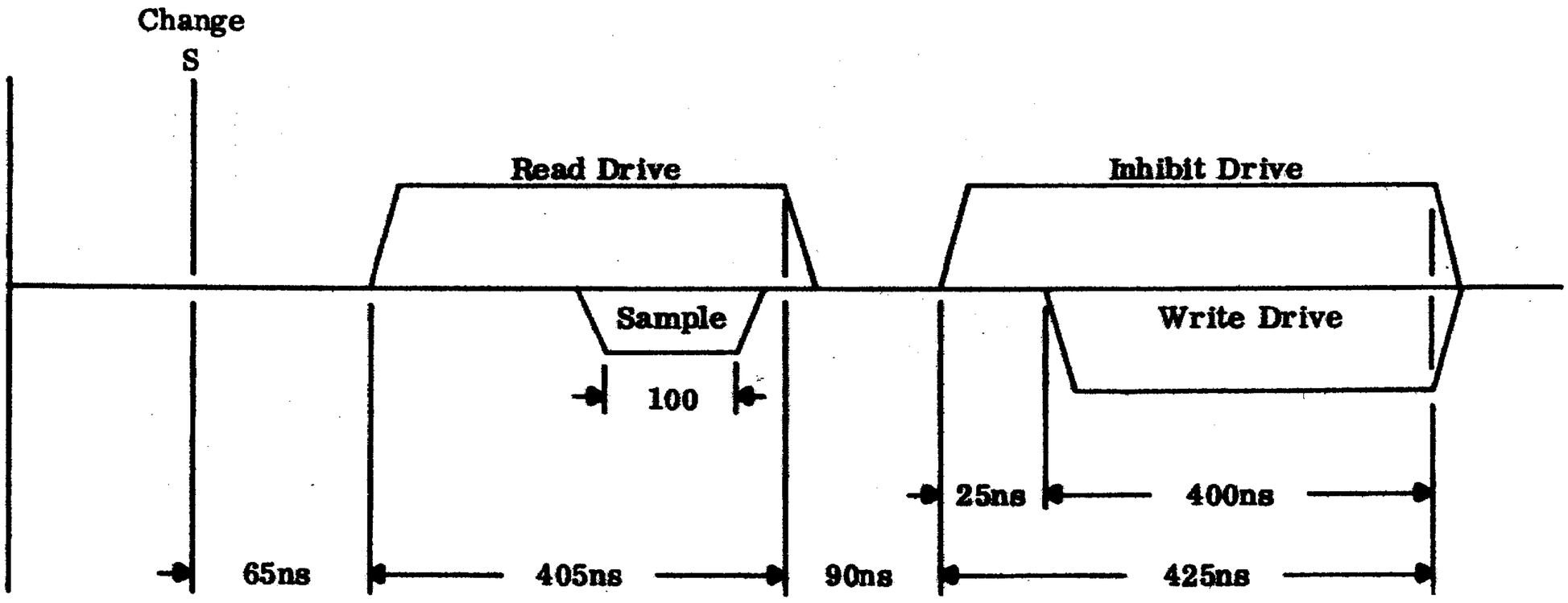


Figure 100. Memory Cycle Timing

The read drive cycle reads the 12 bits at the selected address and produces a one on the sense line for stored one. In doing this, the read drive switches the cores to zero and destroys the information. A stored zero produces a zero on the sense line. The data read out is restored or written in memory during the write drive cycle. The write drive attempts to write all ones for the data word, but the inhibit drive prevents writing a one in those bit positions where a zero should be stored.

The word to be stored is held in an External Write or Restore register. The word in this register for a read memory reference is the word read out on the sense line, but for a write reference it is the word to be stored. External control selects the proper word.

The 12-bit address is available and translated statically during the read, inhibit, and write drive cycles. External control sends Read, Inhibit, and Write Drive signals to the module in the proper time sequence so that drive and inhibit current is applied to the lines as shown in figure 101.

DRIVE CIRCUITS

Translations of the 12-bit address provide 4096 unique codes to select any one core on a plane (actually, same core on 12 planes). Figure 102 shows the relation of the address bits to the storage cycle.

The two drive decks are identical and active coincidentally since both receive the address. The drive (assume horizontal) circuits have separate primary read and write current sources. Both feed eight current transformers (figure 102) through a double primary winding arrangement. These transformers provide eight current sources, each of which feeds one drive line transformer in each of eight groups. The drive transformer secondary winding connects to the ends of the drive line which threads all 12 planes.

There are eight diverter circuits and each feeds eight successive driver transformers. One of the eight diverters is selected to divert current to the proper drive line transformer and its drive line. Therefore, a single current source can be diverted to any of 64 drive lines. Separate translations on three-bit portions of the address select one of eight current source transformers and one of eight diverters to divert current from the source to a drive line. Since both drive decks are turned on at the same time, a coincident current is produced in one core on each plane.

The 64 drive transformers on each deck are so arranged that the 32 odd line transformers are on one edge and the 32 even are on the opposite edge. Successive lines are driven from opposite sides of the deck. The arrangement provides an opposing current relationship with the inhibit drive lines.

CURRENT SOURCE

The primary read or write current sources are large inductances charging chokes connected between the +6v supply and the eight current transformers (figure 102). Current of about 250 ma is available and this is shunted to ground through transistors Q1 (and Q2) and its collector resistor (several in parallel) during nonread or nonwrite times. This load network is provided so that the power supply load is nearly constant at all times. Losses in the current transformer and drive line transformer allow about 200 ma to be delivered to the drive line during the storage cycle.

One of eight current transformers is selected from a 2 by 4 transformer matrix. The upper two bits of the three-bit drive translation select one of four transformer groups, and the lower bit selects one of the two in that group. The Read or Write Drive signal is combined with the one of two selection.

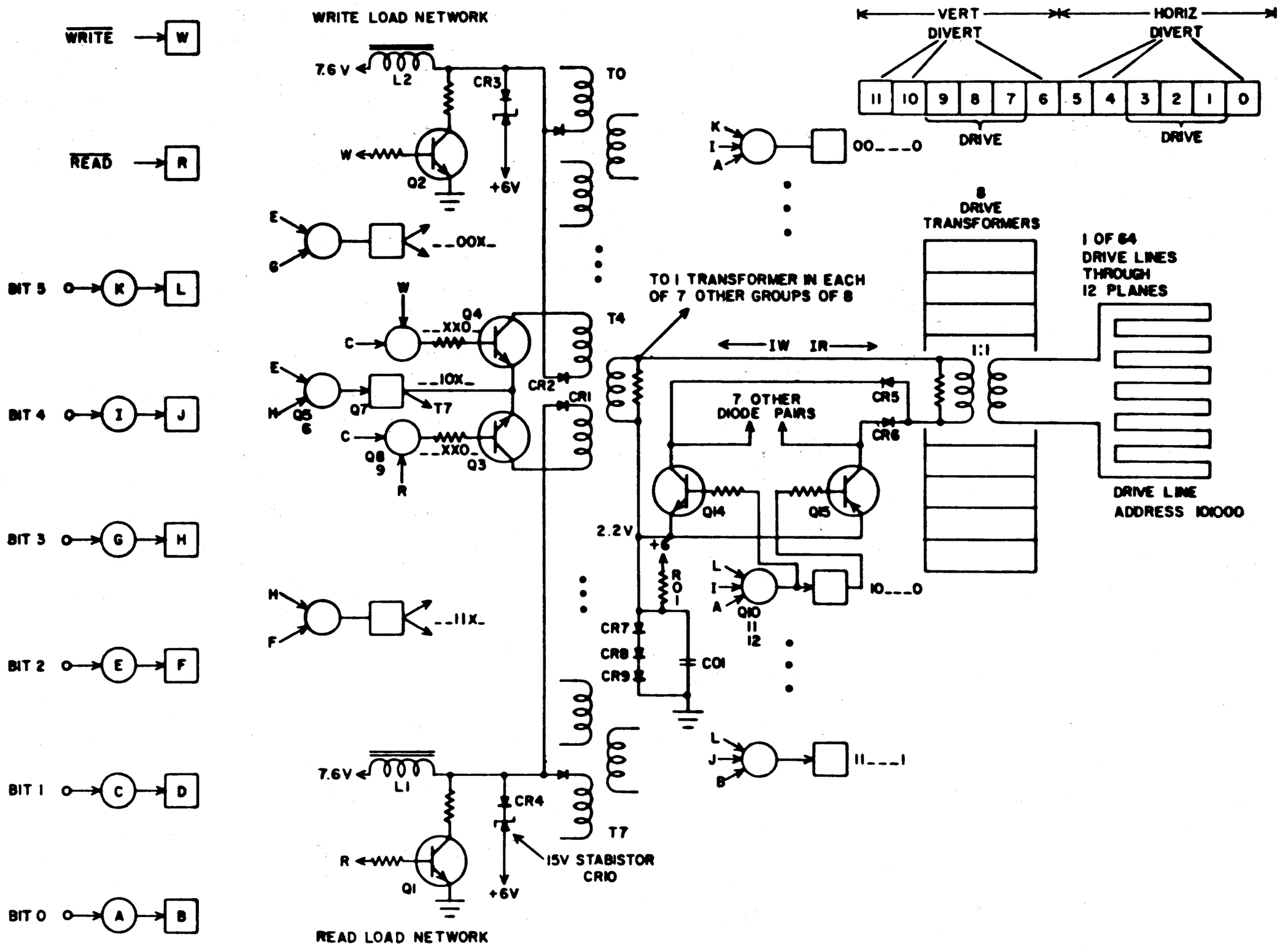


Figure 101. Drive and Inhibit Timing

Figure 101 assumes a 100-binary code in the three-bit drive position and shows that transistors Q5 and Q6 turn off (shown as AND) and Q7 turns on to select transformers T4 and T5 in the one of four group. Transistors Q8 and Q9 turn off and Q3 turns on to select the read side of T4. Read load network transistor Q1 turns off and current flows from ground to the +6v source via transistors Q7 and Q3, transformer T4, diode CR1, and inductor L1.

A high impedance is presented to inductor L1 when read load network transistor Q1 turns off and the voltage level at the junction of diode CR1 and inductor L1 rises rapidly. However, diode CR4 conducts to place stabistor CR10 in the circuit which limits the peak voltage at the junction to 15v. Current loss in T4 is about 25 ma, so 225 ma flow in the secondary and are diverted to a drive transformer.

A voltage is coupled to the other primary winding of T4 when Q3 is conducting. The voltage back-biases diode CR2 to prevent the collector of Q4 from becoming negative with respect to its base. This turns on an unselected circuit and by so doing defeats the selection mechanism.

The circuit operates in a similar manner in the write cycle except that polarities in the current transformer are reversed.

DIVERTER CIRCUITS

The eight diverter circuits divert read and write current to the drive transformers. Each circuit feeds eight drive transformers, one for each current source transformer. The selected current source transformer drives eight lines, but only one of the eight is diverted.

Figure 101 shows one of eight diverter circuits and its selection from the three address bits used to divert. The collectors of Q10, Q11, and Q12 feed the base of the NPN transistor directly and the base of the PNP transistor Q15 indirectly via transistor Q13

which inverts the combined Collector signal. The action establishes positive and negative turn-on signals for the bases of Q14 and Q15. Emitter bias on these two transistors is established by the divider network of R01, C01, and diodes CR7, 8, and 9. The diode threshold is about .7v for a combined drop of about 2.2v and proper emitter bias.

Diode CR5 is forward-biased during read drive to complete the circuit of the selected current source transformer. Diode CR6 is back-biased during read drive and Q15 does not conduct. Current flows from the current transformer through the drive transformer, diode CR5, transistor Q14, and back to the current transformer.

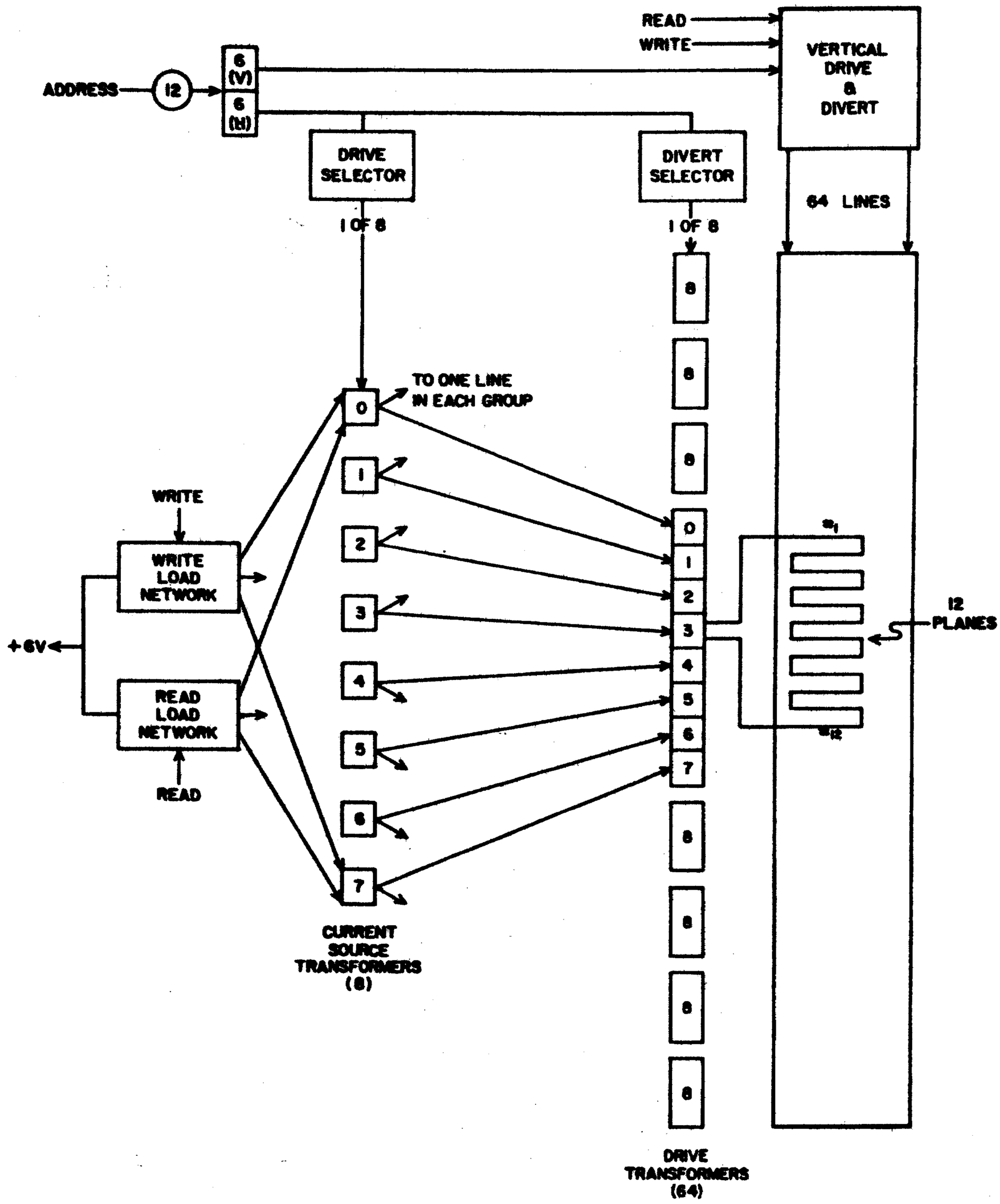


Figure 102. Primary Read/Write Current Sources

Some current is lost in the drive transformer. About 200 ma of the available 225 ma are delivered to the drive line, which appears as a balanced load to the drive transformer secondary. Resistors connected across the current source secondary winding and drive transformer primary winding load the transformers and damp out the transformer ringing produced when the transistors shut off at the end of the cycle.

Diode CR6 is forward-biased and diode CR5 is back-biased during write drive. The circuit operates the same as during read drive except current flow and polarities are reversed. The write drive starts (write load network turned off) 90ns after the read drive stops. The delay period allows the transformers to discharge their stored energy, which prevents a flux buildup or transformer bias condition.

INHIBIT CIRCUITS

The inhibit circuits cancel the effect of write drive and allow zeros to be stored in memory. Inhibiting is done with 100 ma drive current on coincident horizontal and vertical inhibit lines. The resulting 200 ma coincident inhibit current produces a magnetic field in opposition to the field from the 400 ma coincident current write drive which cancels half of the write drive field. The resultant field, or half-amplitude pulse, is not sufficient to switch a core so it remains in the zero state produced by the read drive.

Inhibit drive is plane selective and is turned on when a given plane should store a zero as indicated by the content of the Restore register. The register holds the complement of the word to be stored. Figure 103 shows the opposing current relationship of inhibit and write drive signals in a four-bit plane.

The inhibit lines are arranged in four 16-line groups (figure 104) in vertical and horizontal directions. Two inhibit decks drive vertical lines, and the other two inhibit decks drive horizontal lines. Each deck handles the four groups in six planes. Only one 16 by 16 core area in each plane receives horizontal and vertical inhibit current when a zero is stored.

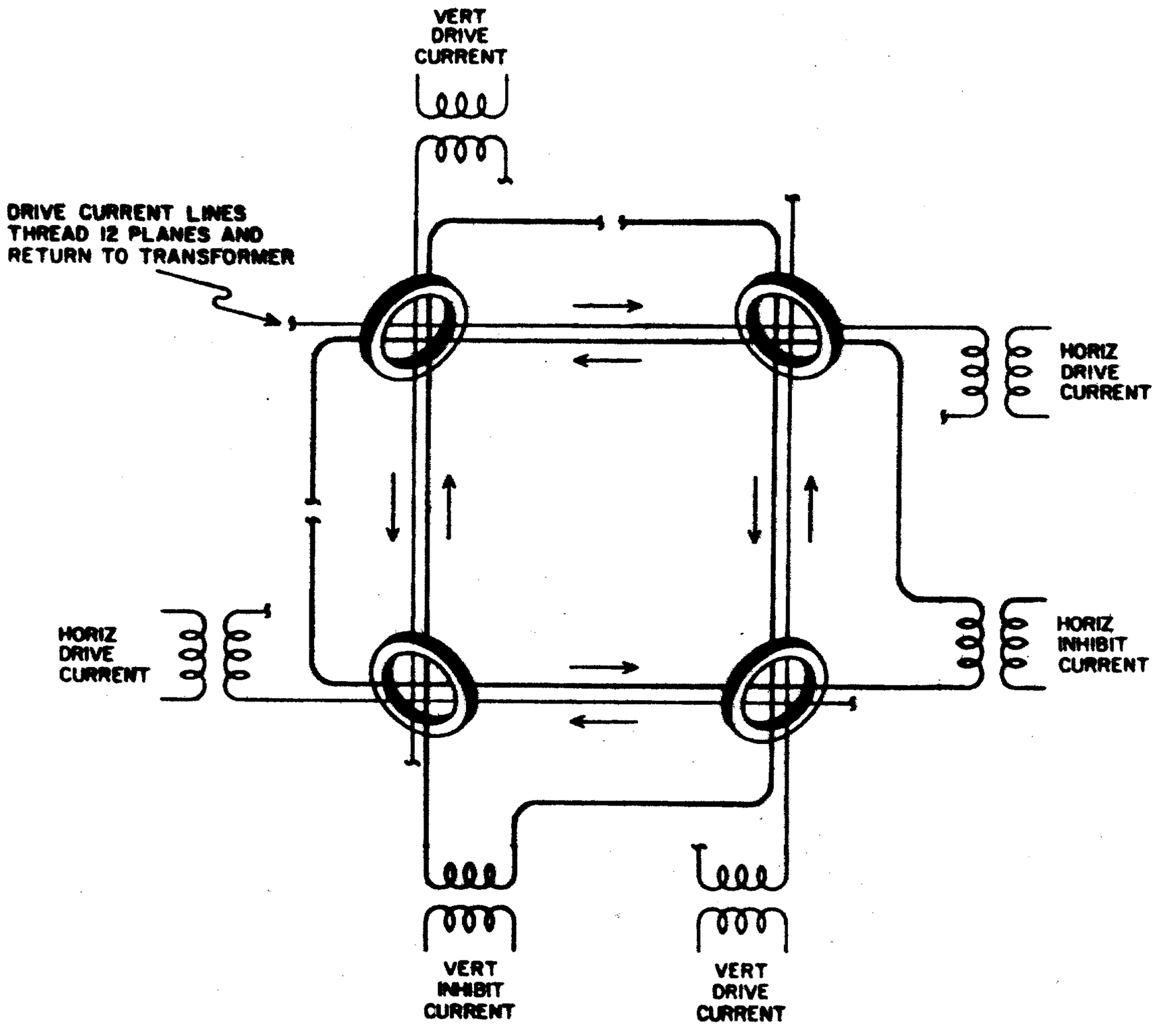


Figure 103. Opposing Current Relationship of Inhibit and Write Drive Signals

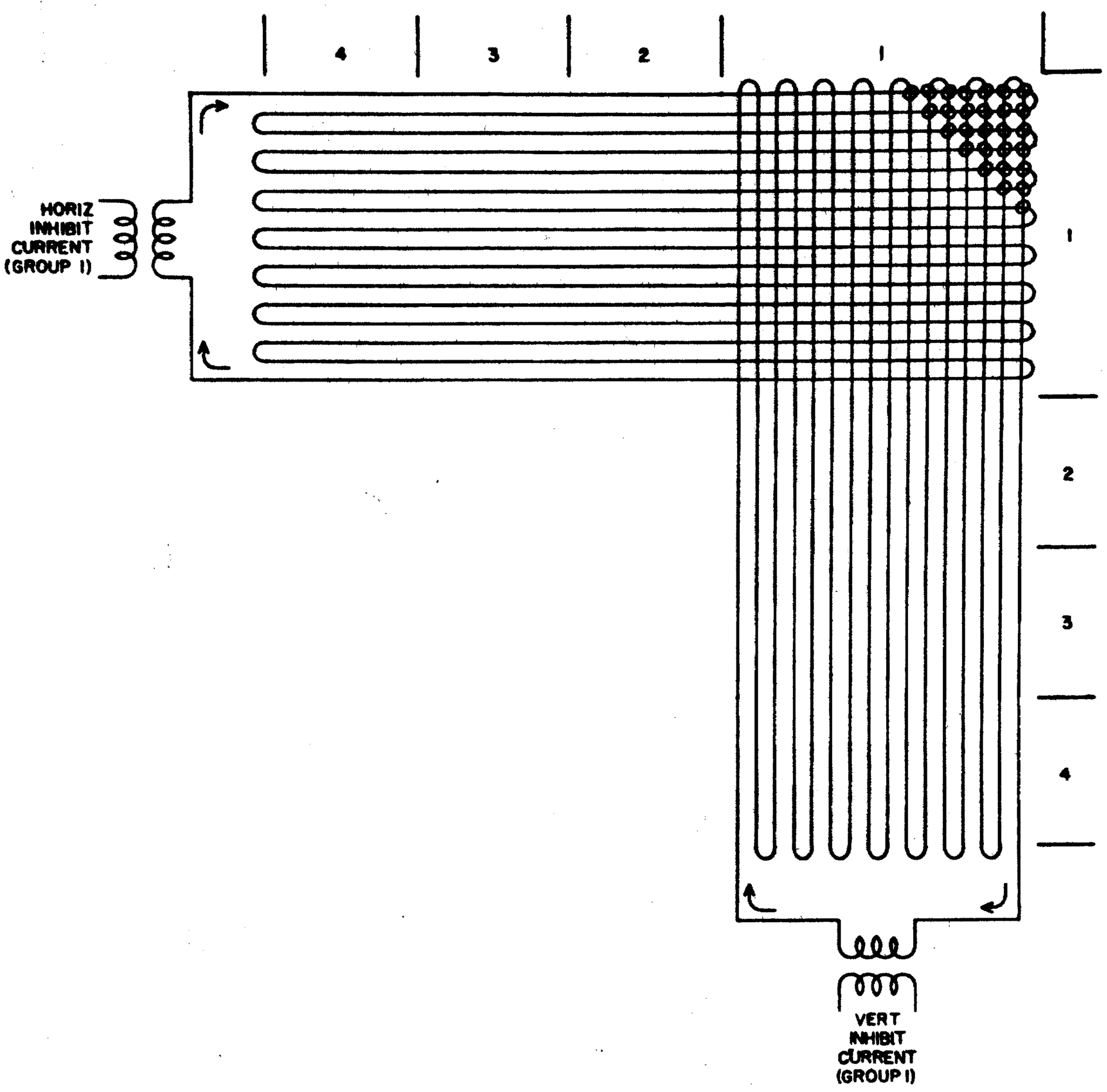


Figure 104. Group 1 Inhibit Lines

The group chosen is a one of four selection from a translation of a two-bit address quantity. Separate two-bit address quantities (figure 105) accommodate horizontal and vertical selections.

Figure 105 shows an inhibit group selection circuit (assume horizontal) activated by two address bits. The circuit forms four unique translations of the bits using diode AND circuits. The output of an AND feeds the base of an emitter follower, which in turn drives one inhibit transformer on each of six planes. An AND circuit turns on its emitter follower when both of its inputs (diode cathodes) are at 1.2v. The anode junction then rises to this potential to turn on the emitter follower.

A separate current source drives the four inhibit transformers required for each plane. Figure 105 shows one of these with the circuit to drive a transformer. During read drive, or the first portion of the storage cycle, the External register (Z) holding the word to be stored is cleared. Slave inverters from its set output provide a +1.2v signal to turn on transistor Q1. Q1 and its collector resistor form a load network for inductor L1. Inhibit current is shunted to ground through R1 and Q1 during noninhibit periods.

The complement of the word to be stored is entered in the Restore register 600ns after the address is received by the storage module. Thus, a zero bit appears in the register as a one, and vice versa. In

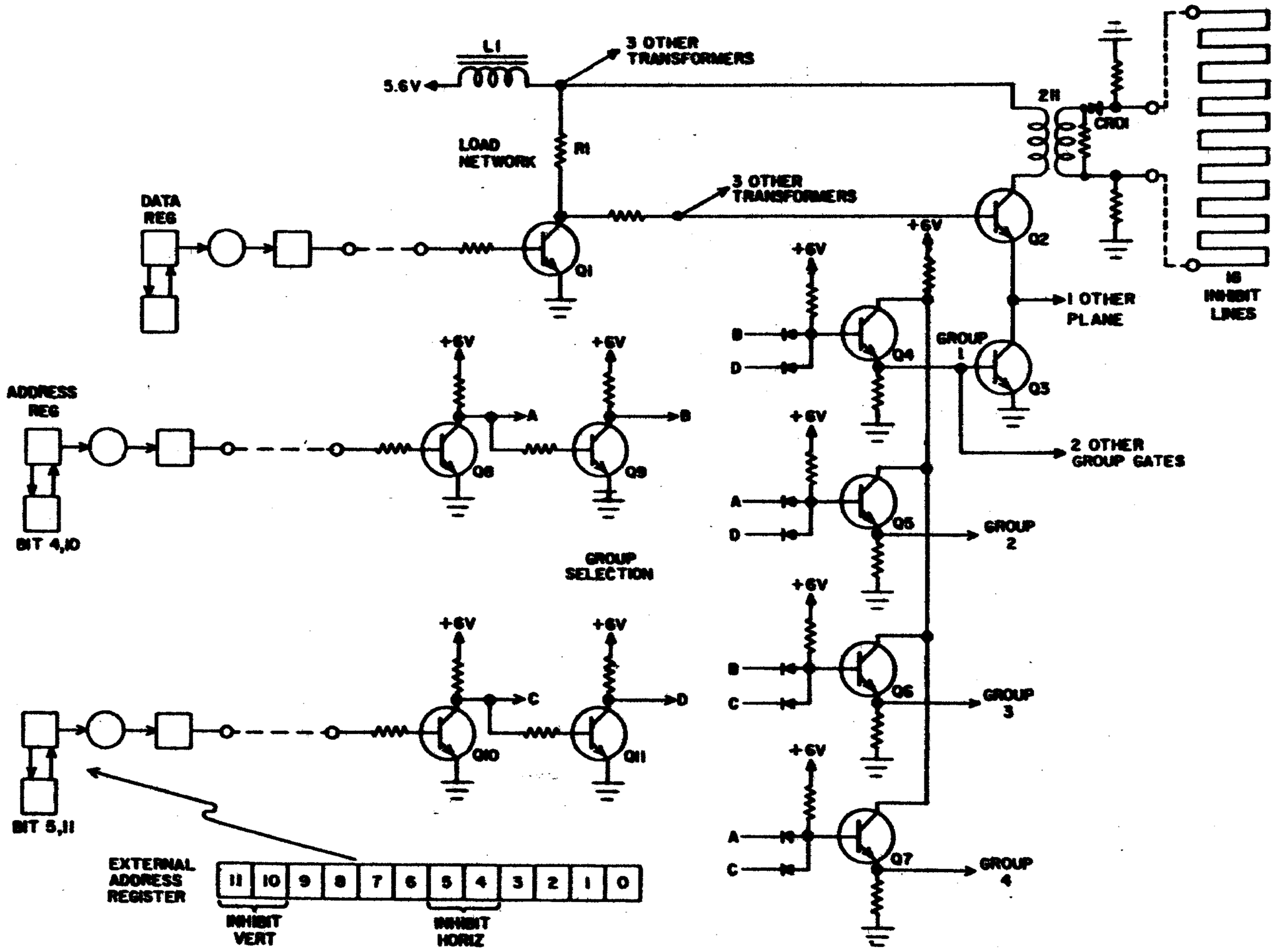


Figure 105. Inhibit Transformer Drive Line Sequence

figure 105, a 0.2v signal turns off load network transistor Q1, and its collector potential rises positive to turn on the base of transistor Q2. The emitter of the latter transistor is already near ground potential because the group selection turned on transistor Q3 at the beginning of the storage cycle. The Q2 collector potential rises rapidly to about 1-v when Q1 turns off. Then it drops off as Q2 starts to conduct current through the transformer primary. A peak current of about 50 ma flows during the voltage swing.

The transformer turns ratio is about 2:1, resulting in a current step-up in the secondary winding to about 100 ma. The secondary voltage turns on diode CR1. The latter conducts the 100 ma current through the inhibit winding which appears as a balanced load. A corresponding 100 ma inhibit current in the vertical plane produces the 200 ma half-amplitude current pulse necessary to cancel the effect of the write drive.

The Z register is cleared 425ns after it is set. This action turns on the 12 load network circuits to stop current flow in the transformer primary and turn off inhibit drive. The collapsing primary field of the inhibit transformer produces a large voltage pulse on the collector of Q2. The polarity of this signal is opposite to the desired polarity of the pulse which appeared at the beginning of the cycle. At this time, the secondary impedance appears quite large to diode CR1, and it cuts off so that a large secondary voltage is generated but no secondary current flows. The voltage produces a flux field in opposition to that produced at the beginning of the inhibit drive. Secondary flux is cancelled to avoid bias shift in the transformer from repeated cycling.

PERIPHERAL PROCESSOR MEMORY

Each of the 10 peripheral and control processors has its own independent core storage unit with a capacity of 4096 12-bit words. Each has its own Address register (S), sense amplifiers, and Restore register (Z). However, the 10 storage units share a common memory cycle path and common paths to and from the barrel (figure 106).

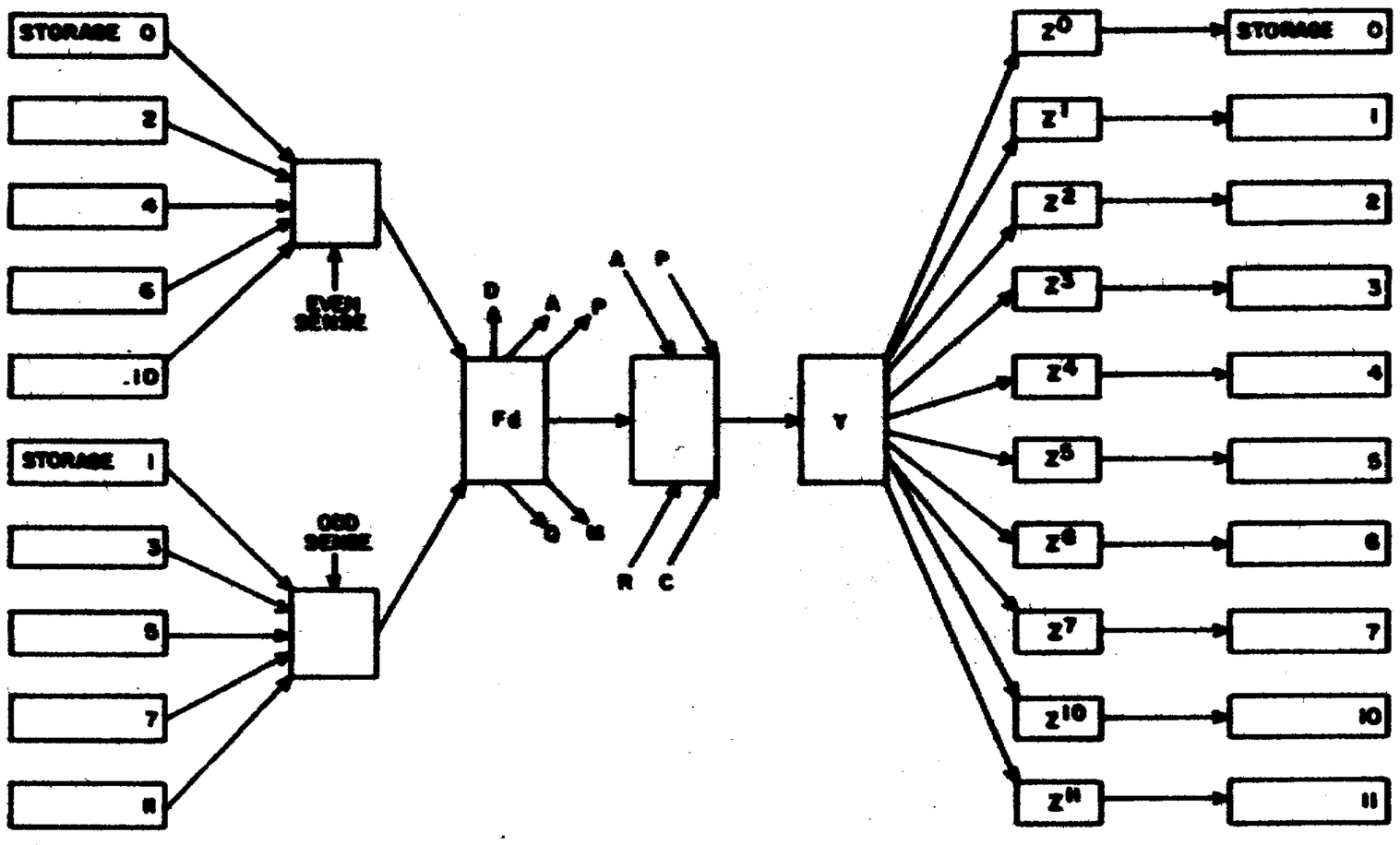


Figure 106. PPU Memory

Each PPU can make one memory reference each major cycle. When no memory reference is called for by the current instruction, address 0000 is read and restored.

STORAGE SEQUENCE CONTROL (SSC)

Timing for memory references is controlled by storage sequence control. Storage sequence control (figure 107) consists of a timing chain of flip-flops gated by clock pulses. As a one passes down the chain, each flip-flop is set for one minor cycle during which it issues commands to the storage logic. This chain reinitiates itself after each cycle and runs continuously. One memory reference is initiated each minor cycle. Since a memory reference requires longer than a minor cycle, the references overlap as shown in figure 107.

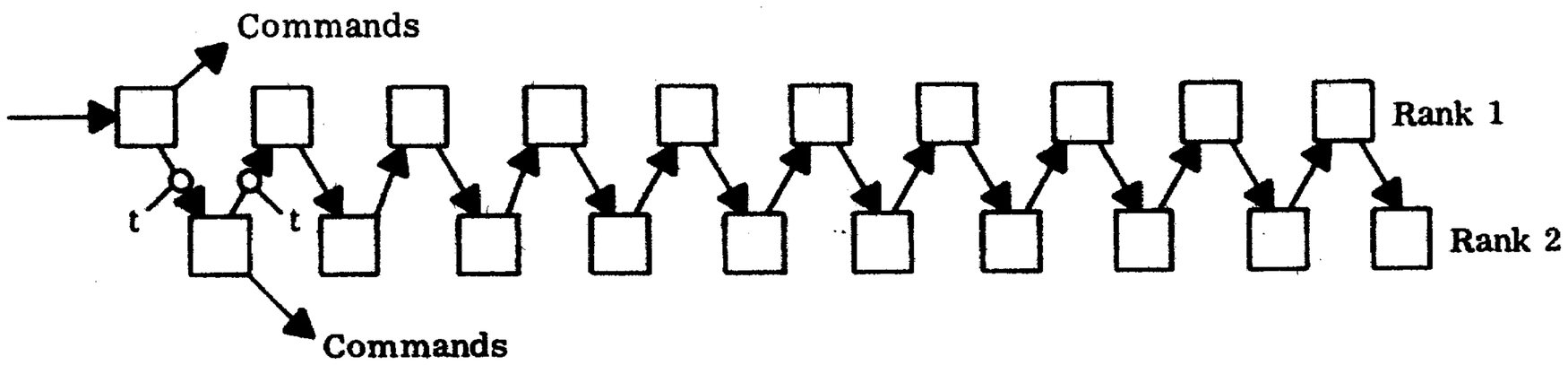


Figure 107. Storage Sequence Control (SSC)

The stages of storage sequence control are numbered according to the processor for which they initiate a memory reference (see figure 108 for typical stage). The commands issued by the first half of a typical stage are listed below.

G → S, Storage a
Clear Z, Storage a+1
Set Z, Storage a+5
Enable Sense, Storage a+7

The second half of stage a issues the following commands.

Read, a
Write, a+5
Stop Read, a+6
Stop Write, a+1

These commands and other signals from storage sequence control define and separate the PPU's.

The reset circuit which reinitiates storage sequence control senses whether stages 0 through 8 are set. If not, stage 0 is reinitiated, normally just after stage 9 has issued its commands (figure 109).

A memory reference is initiated six minor cycles

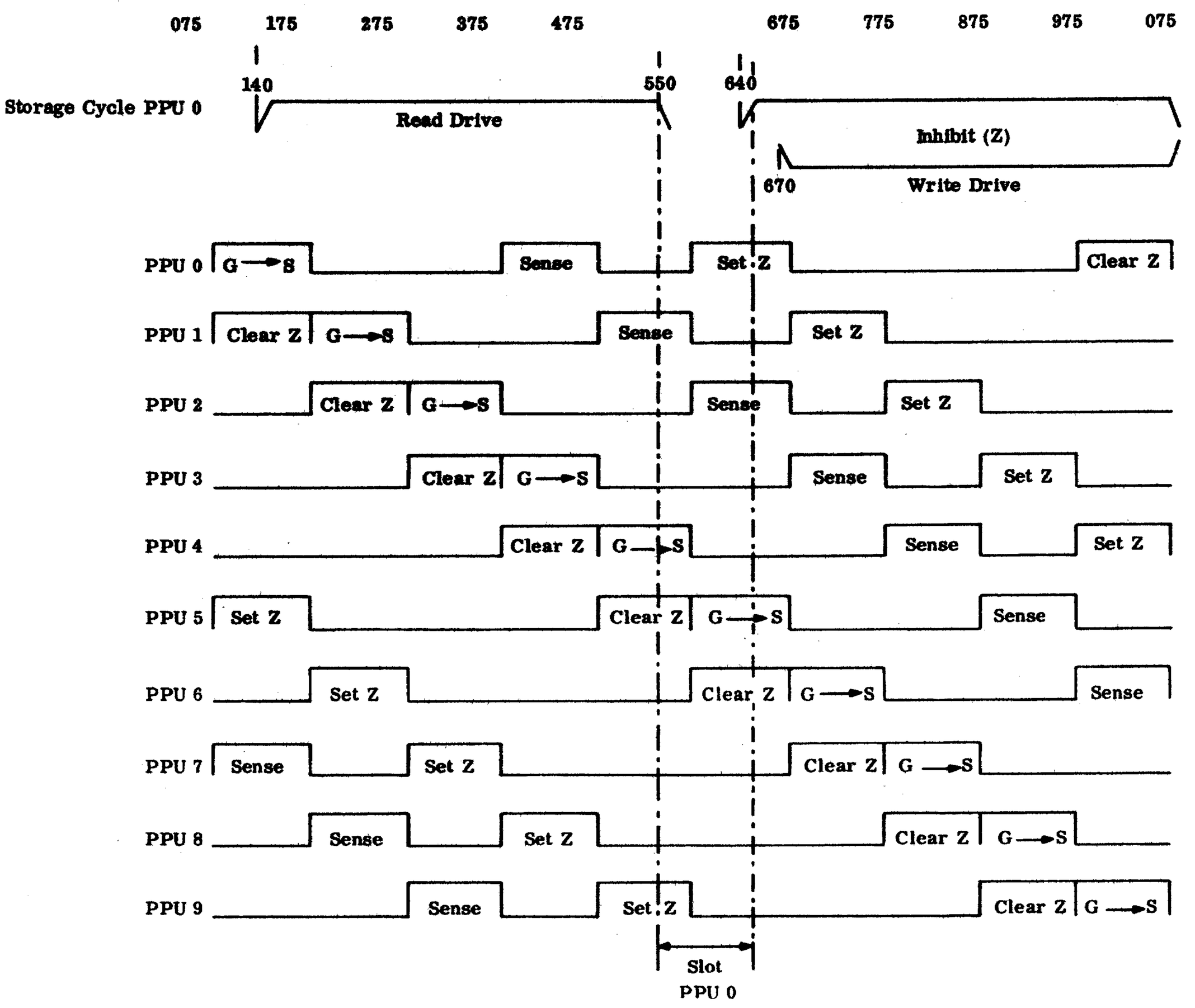


Figure 108. Stages of Storage Sequence Control

before a processor reaches the slot so that information from memory is available at slot time. Therefore, a memory reference for processor 0 (storage 0) is initiated at about the same time that processor 4 enters the slot.

MEMORY CYCLE PATH

The common memory-cycle path used by all processors receives data from the memories via the sense merge (figure 110). Inputs to the sense merge from the sense amplifiers are a logical one (0.2v) when sense is not enabled. When a processor's sense amplifiers are enabled, the output is allowed to go to +0.2v for a sensed one. The AND combination of logical one's from unselected processors, even or odd sense enable, and one bits from the selected processor's sense amplifiers sets the word from memory in the Fd register in the memory cycle path.

The memory cycle path sends information to the barrel, I/O channels, translators, and central write pyramid. It receives information from the barrel, central read pyramid, and I/O channels. Outputs from Fd in the memory cycle path are translated and and used to form commands when K = 00X (RNI trip).

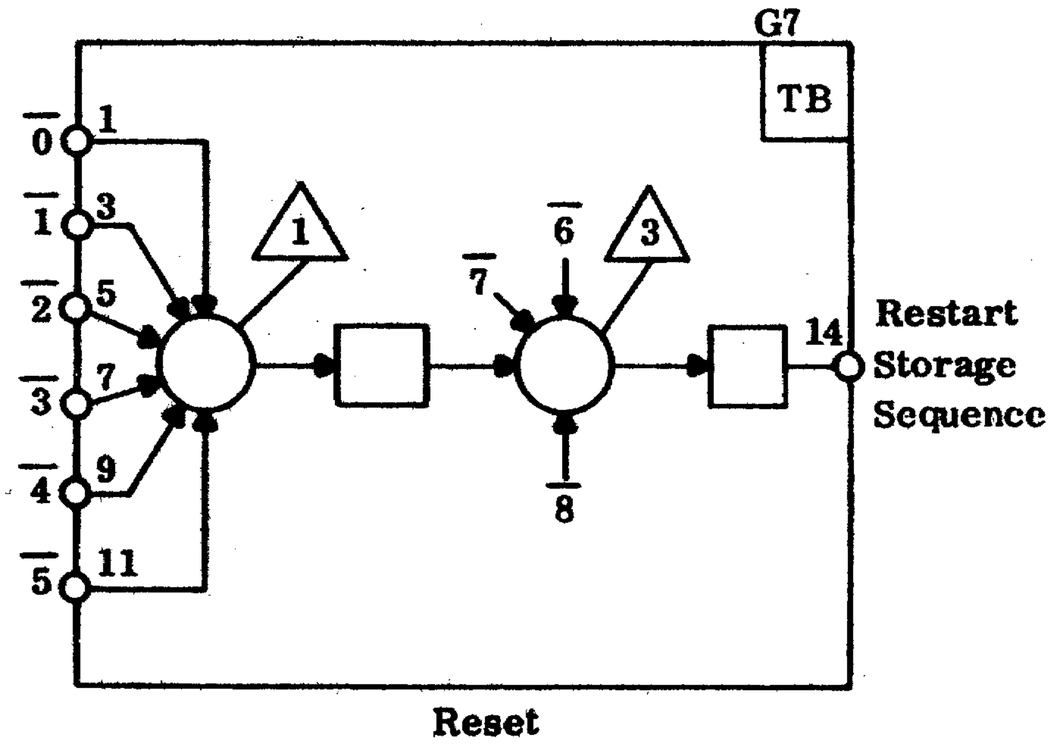


Figure 109. Reset Circuit

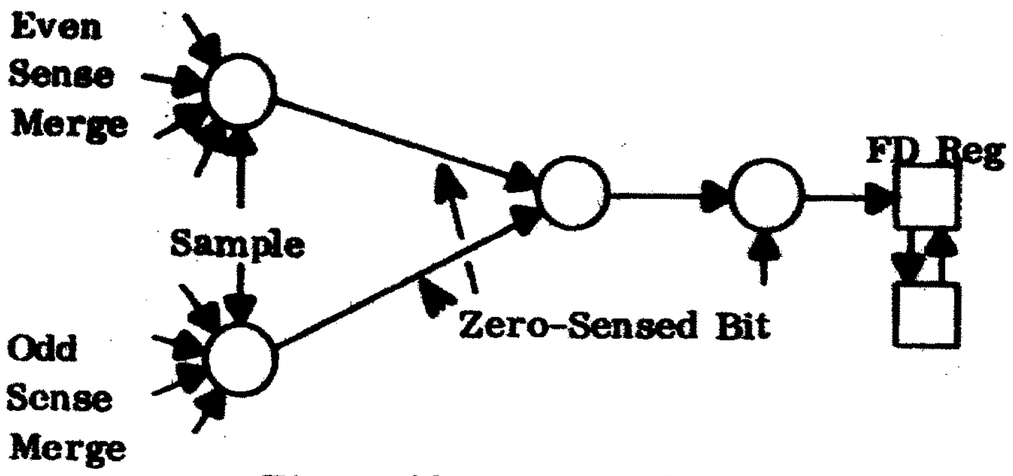


Figure 110. Sense Merge

Information in the memory-cycle path (either the read word or a new word) is fanned out from the Y register to the 10 Z registers. The Set Z signal from storage sequence control gates the complement of the word to be stored into the proper Z register.

POWER

The polarity of the diodes shown in figure 111 is for a positive V (DC voltage). For a negative V the diode would be reversed. These diodes are mounted externally to the transformer.

The circled numbers indicate inputs (1, 2, 3) outputs (4, 5, 6, 7, 8, 9, 10) and test lug (11). The input windings, output windings, and the inductor output are all enclosed in a solid case with no access other than the numbered leads. The output rating of the transformer is fixed by design, but of course varies with input voltage and output loading changes.

Capacitor C (figure 111) is a filter capacitor and is 320 uf at 30 vdc.

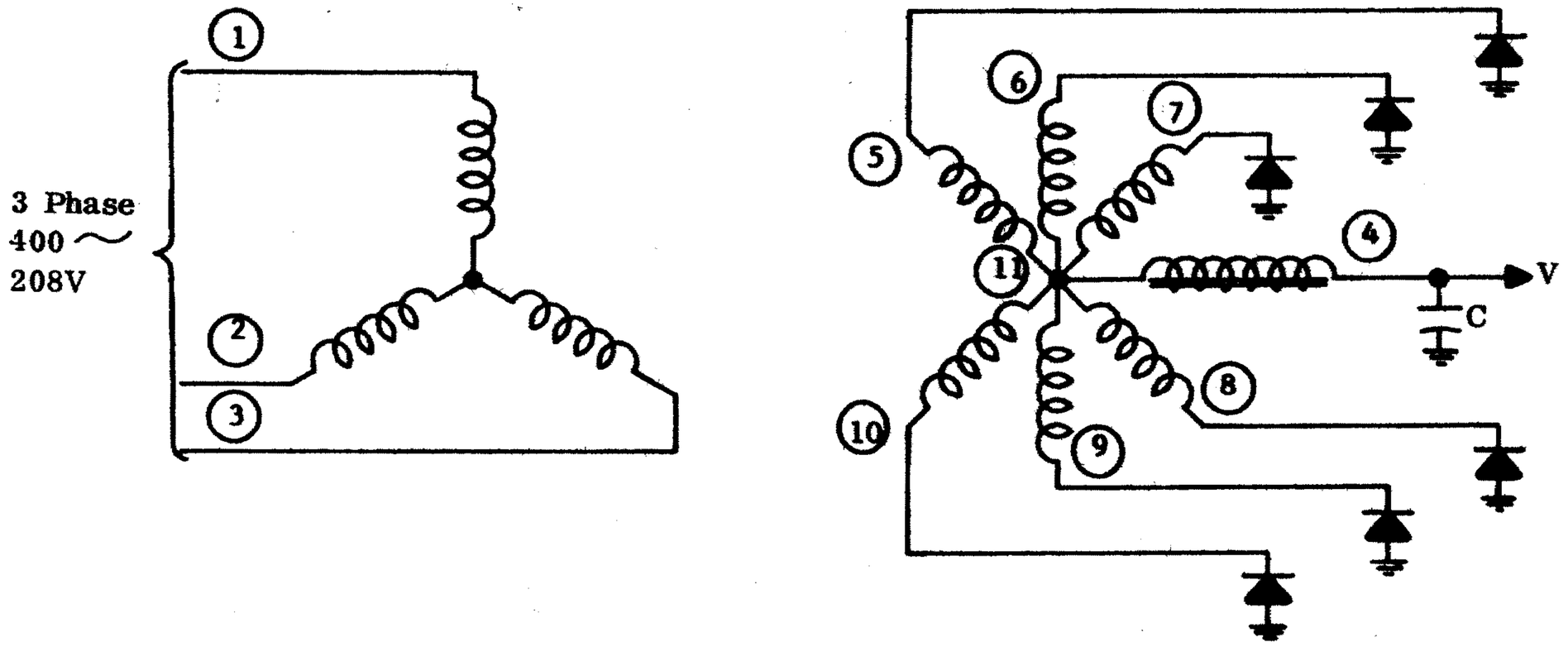


Figure 111. 400 CPS Power Supplies

6000 computers use +5.8 vdc for all normal logic modules. The power enters the module on pin 30 and is filtered by a 0.1 uf capacitor on the module. The storage modules use three voltages: +5.8 vdc for their internal logic, +4.8 vdc for their inhibit supply, and +15 vdc to the current regulator which produces +6.6 vdc for the drive voltage supply. The 5.8v supply is applied directly into the storage module from the bus bar (via the memory buggies). The +4.8v and +6.6v supplies are applied to the storage module via three one-ampere circuit breakers located between the memory buggies.

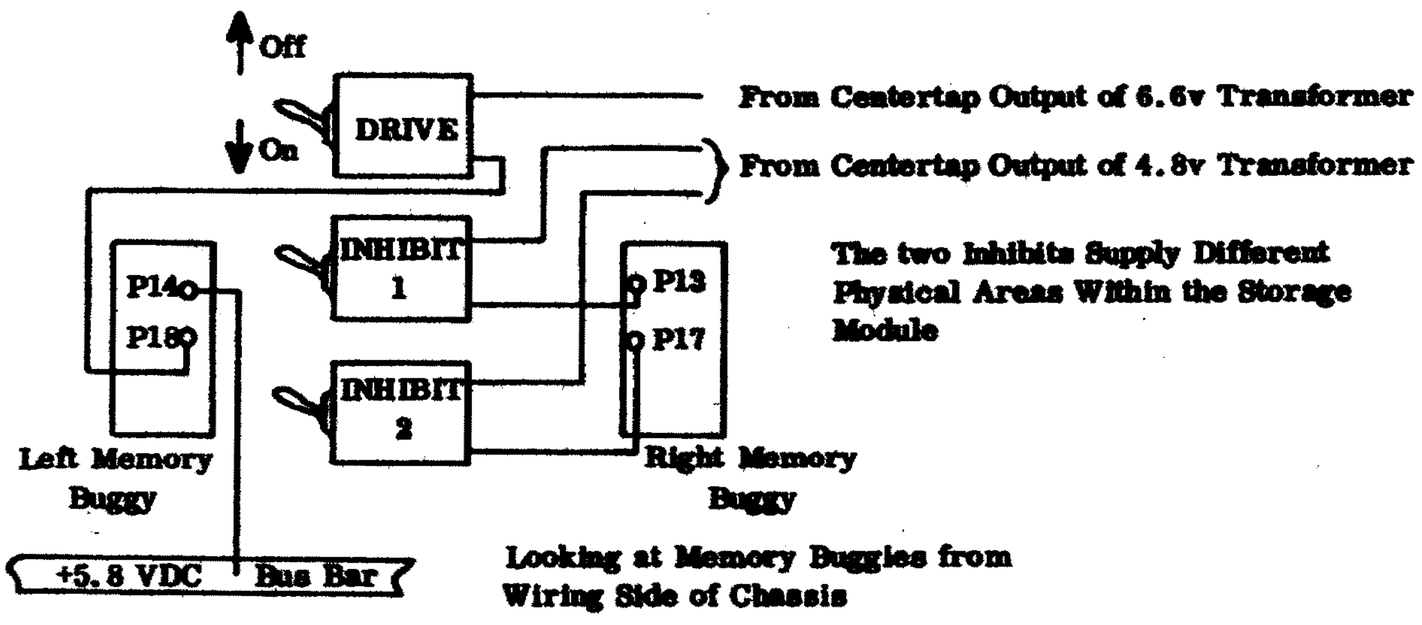


Figure 112. Power Supply Usage

CHAPTER 7

6000 SERIES FLOATING POINT

INTRODUCTION

Floating-point hardware provides the ability to express a number in the general form, $K \bullet B^n$ where:

K = Coefficient
B = Base number
n = Power to which the base number is raised

This concept is opposed to fixed-point hardware where a number is expressed without exponent. Fixed-point hardware places the burden of exponent expression and manipulation upon the software.

To express numbers of the above general form, provision is made to represent four properties: magnitude of the coefficient, sign of the coefficient, magnitude of the exponent (power), and sign of the exponent.

The base number, B, is assumed to be two. Hence there is no need to express the base number. In other words, the computer assumes the exponent to be a power of two. The 6000 series 60-bit floating-point format is shown in figure 113.

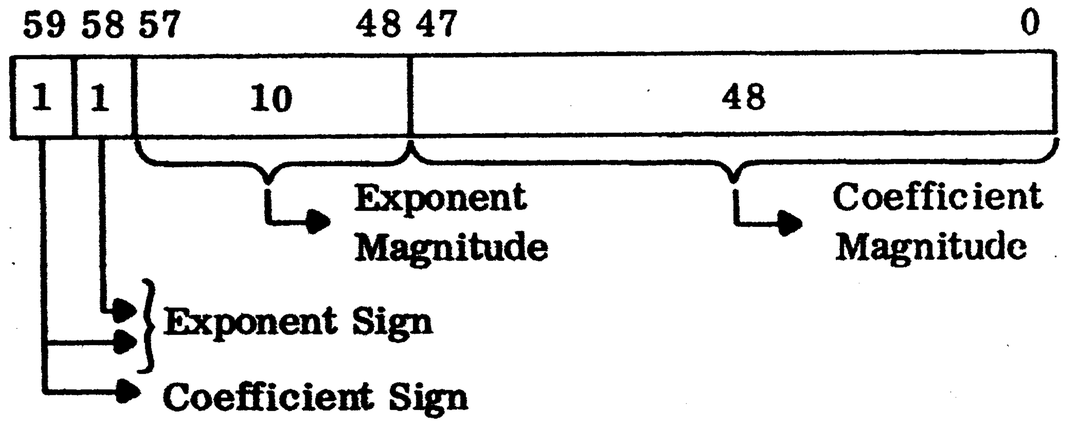


Figure 113. 6000 Series Floating-Point Format

The lower 48-bits are reserved for expressing the magnitude of the coefficient. The computer assumes the binary point to be to the right of the coefficient, thereby providing a 48-bit integer coefficient which is equivalent to about 15 decimal digits. Bit 2^{59} is used to express the sign of the coefficient. Table 16 gives the rules that apply to signed coefficients.

Table 16. Signed Coefficient Rules

Sign	2^{59} =	Magnitude (Bits 0 through 47)
Positive	0	True (uncomplemented) form
Negative	1	Complement form

Bits 48 through 57 are used to express the magnitude of the exponent. The absolute value of the exponent can be as large as 1777_8. The sign of the exponent is expressed by the combination of bits 2^{58} and 2^{59}, according to these rules. If $2^{58} \neq 2^{59}$, the exponent sign is positive, if $2^{58} = 2^{59}$, the exponent sign is negative.

Table 17 summarizes the configuration of bits 2^{58} and 2^{59} and the implications, regarding signs, of the possible combinations.

Table 17. Description of Bits 2^{58} and 2^{59}

2^{59}	2^{58}	Coefficient Sign	Exponent Sign
0	1	Positive	Positive
0	0	Positive	Negative
1	0	Negative	Positive
1	1	Negative	Negative

PACKING

Packing refers to the conversion of numbers in the form $K \bullet B^n$ into floating-point format. The rules outlined below apply to this process.

A. Positive Coefficient
   1. Place the coefficient magnitude in positions 47 through 0.
   2. Make bit 59 equal to zero.
   3. If Exponent is Positive
      a. Make bit 58 equal to one.
      b. Place the exponent magnitude in positions 57 through 48.
   4. If Exponent is Negative
      a. Make bit 58 equal to zero.
      b. Complement the exponent magnitude and place in positions 57 through 48.

B. Negative Coefficient
   1. Pack according to the four rules for positive coefficients.
   2. Complement all 60 bits.

Example 28

Positive Coefficient and Positive Exponent

Problem: Pack the number $+3427 \bullet 2^{+26}$

Solution:

1	Rule A1.	Coef Mag	= ----0000000000003427
2	Rule A2.	Coef Sign	= 0---0000000000003427
3	Rule A3a.	Exp Sign	= 2---0000000000003427
4	Rule A3b.	Exp Mag	= 20260000000000003427

The result of step 4 yields the packed quantity.

Example 29

Positive Coefficient and Negative Exponent

Problem: Pack the number $+3427 \bullet 2^{-26}$

Solution:

1	Rule A1.	Coef Mag	= ----0000000000003427
2	Rule A2.	Coef Sign	= 0---0000000000003427
3	Rule A4a.	Exp Sign	= 0---0000000000003427
4	Rule A4b.	Exp Mag	= 17510000000000003427

The result of step 4 yields the packed quantity.

Example 30

Negative Coefficient and Positive Exponent

Problem: Pack the number $-3427 \bullet 2^{+26}$

Solution:

1	Rule B1.	Use rules A1 through A4 first	
2	Rule A1.	Coef Mag	= ----0000000000003427
3	Rule A2.	Coef Sign	= 0---0000000000003427
4	Rule A3a.	Exp Sign	= 2---0000000000003427
5	Rule A3b.	Exp Mag	= 20260000000000003427
6	Rule B2.	Complement	= 5751777777777774350

The result of step 6 yields the packed quantity.

Example 31

Negative Coefficient and Negative Exponent

Problem: Pack the number $-3427 \bullet 2^{-26}$

Solution:

1	Rule B1.	Use rules A1 through A4 first	
2	Rule A1.	Coef Mag	= ----0000000000003427
3	Rule A2.	Coef Sign	= 0---0000000000003427
4	Rule A4a.	Exp Sign	= 0---0000000000003427
5	Rule A4b.	Exp Mag	= 17510000000000003427
6	Rule B2.	Complement	= 6026777777777774350

The result of step 6 yields the packed quantity.

In examples 30 and 31, check the sign of the coefficient and exponent as given by the packed quantity. In example 30, the upper octal is 5 or 101_2. According to table 15, this implies a negative coefficient and a positive exponent which are the correct signs of the original number. In example 31, the upper octal is 6 or 110_2. According to table 15 this implies a nega-

tive coefficient and a negative exponent which are the correct signs of the original number.

Although the above process is not a difficult one, it is somewhat cumbersome. A short-cut method of packing exponents can be derived by considering the representation of negative and positive zero exponents. Assuming a positive coefficient, zero exponents are packed as follows.

Positive Zero Exponent = 2000X----------X
Negative Zero Exponent = 1777X----------X

Since positive exponents are expressed in true form, start with a bias of 2000 (positive zero) and add the magnitude of the exponent. The range of positive exponents is 0000 through 1777 or, in packed form, 2000 through 3777.

Negative exponents are expressed in complement form. Start with a bias of 1777 (negative zero) and subtract the magnitude of the exponent. The range of negative exponents is -0000 through -1777 or, in packed form, 1777 through 0000.

The rules for packing can now be outlined in the following simplified manner.

A. Positive Coefficient
   1. Place the coefficient magnitude in positions 47 through 0.
   2. If the exponent is positive, add the exponent magnitude to 2000.
   3. If the exponent is negative, subtract the exponent magnitude from 1777.

B. Negative Coefficient
   1. Pack according to the rules for positive coefficients.
   2. Complement all 60 bits.

UNPACKING

As the name implies, unpacking is the process opposite of packing. It refers to the conversion of floating-point formats into numbers of the form: $K \bullet B^n$. The rules outlined below apply to this process.

A. Check Sign of Coefficient (Bit 59)
   1. If positive, take the lower 48 bits as the coefficient magnitude.
   2. If negative, complement all 60 bits and take complemented lower 48 bits as the coefficient magnitude.

B. Check Complemented Upper 12 Bits.
   1. If in the range 2000 through 3777, subtract 2000 (bias) to obtain the magnitude of the positive exponent.
   2. If in the range 0000 through 1777, subtract from 1777 (bias) to obtain the magnitude of the negative exponent.

These rules can be proven by working examples 28 through 31 in reverse to obtain the original mathematical expression.

RANGE DEFINITIONS

The preceding explanation presented the format and conversion methods of 6000 series floating point. The 6000 computers provide the capability of representing a 48-bit integer coefficient multiplied by 2 raised to a power in the range $\pm 1777_8$. In octal this is expressed as $7777777777777. \bullet 2^{\pm 1777}$. The decimal equivalent is $281474976710655. \bullet 10^{\pm 307.95369}$.

Overflow

A result with an exponent so large that it reaches or exceeds the upper limit of 3777 (positive) or 4000 (negative) is treated as an infinite quantity. In the case of positive infinity, a coefficient of all zeros and an exponent of octal 3777 are packed. For negative infinity, a coefficient of all zeros and an exponent of octal 4000 are packed. An optional exit is provided when an infinite operand is detected because its use may generate further error conditions.

Underflow

The exponent of a result which is less than the lower limit of 0000_8 (positive) or 7777_8 (negative) is treated as a zero quantity. This condition causes a zero exponent and coefficient to be packed. No exit is provided for underflow. A result with an exponent of 0000_8 and a coefficient that is not zero is a nonzero quantity and is packed with 0000_8 exponent and the nonzero coefficient.

Indefinite Operands

Use of infinity, zero, or indefinite operands may produce an indefinite result. An exponent of 1777_8 and a zero coefficient are packed in this case. An optional exit is provided. Therefore, octal exponent configurations of 1777, 3777, 4000, and 6000 have special meaning and should not be used in representing numbers in floating point.

Summary

In summary, the special operand forms in octal are listed below.

- 3777X —— X Positive infinity
- 4000X —— X Negative infinity
- 1777X —— X Positive indefinite
- 6000X —— X Negative indefinite
- 00000 —— 0 Positive zero
- 77777 —— 7 Negative zero

Whenever an operand in one of these six special forms is used as a source operand, only the following octal words can occur as results.

- 37770 — 0 Positive infinity
- 40000 — 0 Negative infinity
- 17770 — 0 Positive indefinite
- 00000 — 0 Positive zero

Note that in these cases, the 48 least significant bits of the result are zeros. For a detailed list of infinite, zero, and indefinite forms, refer to the 6400/6600 Reference Manual, publication number 60100000.

SINGLE/DOUBLE PRECISION

The 6000 series Floating Point Add, Subtract, and Multiply instructions are capable of producing single or double precision results. Single precision results may be rounded or unrounded; double precision op codes always return unrounded results. Floating Point Divide instructions generate only rounded or unrounded single precision results.

Floating Add Operations

Before addition of floating-point numbers takes place, the exponents are equalized by shifting the coefficient of the smaller exponent to the right a number of places given by the difference of the two exponents. The shifted and unshifted coefficients are added and a 96-bit sum is generated as illustrated in example 32. Neither coefficient is shifted if the original exponents are equal.

Example 32

Given

FX6 = X1 + X2
(X1) = 2072 4273000012340772
(X2) = 2057 5230000023457610

Add logic forms

X1 Unshifted	=	4273000012340772.
X2 Right Shift 15 places	=	523000000234.57610
96-Bit Result	=	427305231234 1226.5761000000000000

In effect, right shifting the coefficient of the smaller exponent (which decreases the coefficient magnitude) increases that exponent to equal the larger. The exponent of the 96-bit sum is the larger of the two original exponents, and the binary point in the example is correctly positioned at the center of the result. Since unrounded single precision is specified (FX6), the upper 48-bits of the sum and the larger exponent are taken as the final result: (X6) = 2072 4273052312341226.

If double precision had been specified (DX6), the final result would have been the lower 48-bits of the sum and the larger exponent minus 60_8 (48_{10}): (X6) = 2012 5761000000000000.

Packing the lower 48 bits of the sum effectively moves the binary point 48 places to the right. This move increases the coefficient magnitude. To compensate, the exponent is decreased by 48 (60_8) during double precision.

If rounded single precision were specified (RX6), a one would be placed below bit 2^0 of the unshifted operand before adding. This, in effect, adds one half and causes a carry into bit 2^0 if the fractional portion of the shifted operand is greater than or equal to one half. To illustrate, the original example with round specified is shown in example 33.

Example 33

Add logic forms

X1 Unshifted	=	4273000012340772.4
X2 Right Shift 15 places	=	523000000234.57610
96-Bit Result	=	427305231234 1227.1761000000000000

The Final result in this case is (X6) = 2072 4273052312341227

Floating Multiply Operations

Multiplication of floating point numbers requires the addition of the exponents and the multiplication of two 48-bit coefficients. A coefficient product of 96 bits, with the binary point below bit 2^0, is formed. If both original operands are normalized, the result also is normalized as shown in example 34.

Example 34

Given

FX7 = X3 * X4
(X3) = 2023 4253000000000002
(X4) = 2027 5300000000000014

Multiply logic forms

X3 4253000000000002.
X4 5300000000000014.
96-Bit Result = 272271000000007460400000000030.

Since this result is not normalized, it is left shifted one place.
LS one place = 564562000000001714100000000060
2131 5645620000000020.

Since single precision was specified (FX7), the upper 48 bits of the result will be packed as the final coefficient. This effectively moves the binary point to the left 48 places and by so doing decreases the magnitude of the coefficient. An octal 60 is added to the sum of the exponents. To compensate for the left shift required to normalize, one is subtracted from the exponent. The single precision result is: (X7) = 2131 5645620000000017.

NOTE

Final exponent = 2023 + 2027 + 60 - 1 = 2131

If double precision had been specified (DX7), the lower 48 bits of the product and the sum of the exponents minus one (compensating for the left shift to normalize) would be taken as the result: (X7) = 2051 1410000000000060.

If rounded single precision had been specified (RX7), a one would be added to bit 2^{46} of Xk (X4 in this example) on the first iteration of the multiply step. A rounded, 96-bit coefficient product would be generated and the upper 48 bits, along with the sum of the exponents plus 57_8, would be taken as the result: (X7) = 2131 5645620000000020.

In the above example one half round occurred because the 96-bit coefficient was left shifted one place to normalize. If the product had been normalized already, the left shift would not have occurred. The result would then have been rounded by one fourth (since the round bit was entered in position 2^{46}) and the final exponents would all be greater by one.

Floating Divide Operations

Division of floating-point numbers requires the subtraction of the exponents and the division of two 48-bit coefficients. Double precision division is not provided and a remainder cannot be retrieved. Since the divide hardware produces a quotient in the range 1.7777777777777777 through 0.0000000000000000, the ratio of Xj to Xk must always be less than two to one. If this requisite is not met, the resulting quotient will be meaningless.* Under these conditions an indefinite operand (1777 0 —— 0) is forced.

If the quotient is of the form 0.X----------X (the ratio of Xj to Xk is less than one to one), the lower 48 bits of a 49-bit quotient are taken as the coefficient of the result as shown in example 35.

Example 35

Given

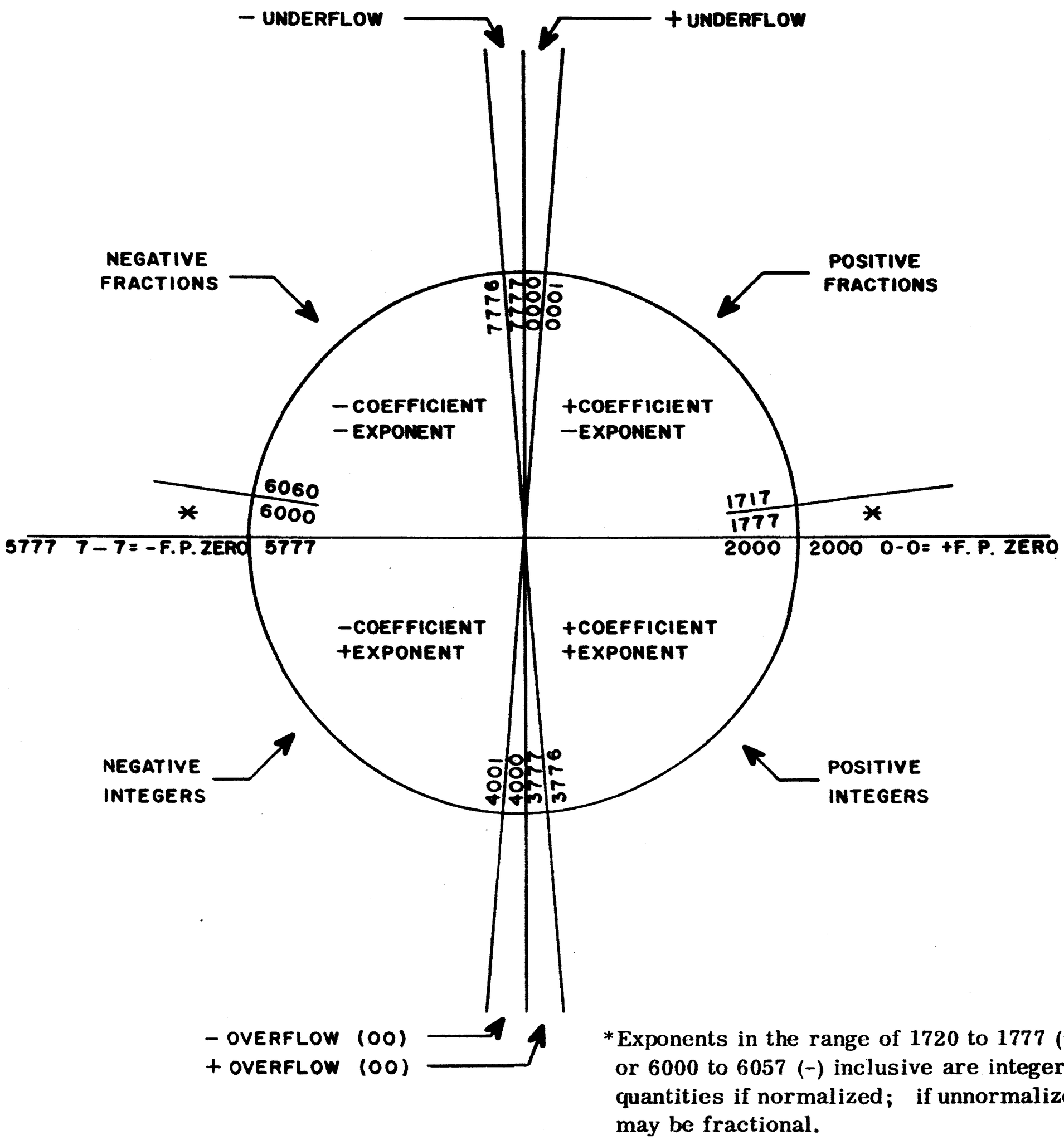
FX6 = X4 / X5
(X4) = 2057 4400000000000015
(X5) = 2032 6000000000000000

Divide logic forms

$$\frac{4400000000000015}{6000000000000000} = 0.6000000000000002$$

*It is suggested that the Xj and Xk operands always be normalized before executing a Divide instruction. Normalizing eliminates the possibility of the dividend/divisor ratio (Xj/Xk) being greater than or equal to two to one.

The machine treats as underflow and packs all zeros for either:
0000 X — X or
7777 X — X



*Exponents in the range of 1720 to 1777 (+) or 6000 to 6057 (-) inclusive are integer quantities if normalized; if unnormalized, may be fractional.

Figure 114. 6000 Floating-Point Exponent Chart

Packing into floating-point format effectively moves the binary point 48 places to the right which increases the coefficient magnitude. Therefore, 60_8 is subtracted from the difference of the exponents and the final result is: (X6) = 1744 6000000000000002.

NOTE

Final exponent = 2057 - 2032 - 60 = -33 = 1744

If the quotient is of the form 1.X----------X (the ratio of Xj to Xk is one to one or greater, but less than two to one), the upper 48 bits of a 49-bit quotient are taken as the coefficient of the result as shown in example 36.

Packing into floating-point format effectively moves the binary point 47 places to the right. Therefore, 57_8 is subtracted from the difference of the exponents and the final result is: (X7) = 1711 7000000000000000.

NOTE

Final exponent = 2016 - 2025 - 57 = -66 = 1711

Example 36

Given

FX7 = X1 / X2

(X1) = 2016 7000000000000000

(X2) = 2025 4000000000000000

Divide logic forms

$$\frac{7000000000000000}{4000000000000000} = 1.6000000000000000$$

NONSTANDARD FLOATING-POINT ARITHMETIC

Table 18 is a tabulation of operations (add, subtract, multiply, divide) using various combinations of operands.

Table 18. Operands for Nonstandard Floating-Point Arithmetic

KEY:

OPERANDS	RESULTS
+0 = 0000 X — X	0 = 0000 0 — 0
-0 = 7777 X — X	IND = 1777 0 — 0
+ ∞ = 3777 X — X	+ ∞ = 3777 0 — 0
- ∞ = 4000 X — X	- ∞ = 4000 0 — 0
+IND = 1777 X — X	
-IND = 6000 X — X	
W = Any word except ± ∞ , ±IND	
N = Any word except ± ∞ , ±IND, or ±0	

ADD

Xi = Xj + Xk

(Instructions 30, 32, 34)

		Xk			
		W	+ ∞	- ∞	±IND
Xj	W	-	+ ∞	- ∞	IND
	+ ∞		+ ∞	IND	IND
	- ∞			- ∞	IND
	±IND				IND

SUBTRACT

Xi = Xj - Xk

(Instructions 31, 33, 35)

		Xk			
		W	+ ∞	- ∞	±IND
Xj	W	- ∞	- ∞	+ ∞	IND
	+ ∞	+ ∞	IND	+ ∞	IND
	- ∞	- ∞	- ∞	IND	IND
	±IND	IND	IND	IND	IND

Table 18. Operands for Nonstandard Floating-Point Arithmetic (Cont)

MULTIPLY
Xi = Xj * Xk
(Instructions 40, 41, 42)

		Xk						
		+N	-N	+0	-0	+ ∞	- ∞	±IND
Xj	+N	-	-	0	0	+ ∞	- ∞	IND
	-N		-	0	0	- ∞	+ ∞	IND
	+0			0	0	IND	IND	IND
	-0				0	IND	IND	IND
	+ ∞					+ ∞	- ∞	IND
	- ∞						+ ∞	IND
	±IND							IND

DIVIDE
Xi = Xj / Xk
(Instructions 44, 45)

		Xk						
		+N	-N	+0	-0	+ ∞	- ∞	±IND
Xj	+N	-	-	+ ∞	- ∞	0	0	IND
	-N	-	-	- ∞	+ ∞	0	0	IND
	+0	0	0	IND	IND	0	0	IND
	-0	0	0	IND	IND	0	0	IND
	+ ∞	+ ∞	- ∞	+ ∞	- ∞	IND	IND	IND
	- ∞	- ∞	+ ∞	- ∞	+ ∞	IND	IND	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

CHAPTER 8

REFRIGERATION SYSTEM

INTRODUCTION

The cooling system uses the principles of common mechanical refrigeration systems to dissipate component heat from the logic and storage modules. Chassis refrigerant temperature is maintained at 60° to 65° F. The cooling system is explained in detail following some explanatory comments on principles of refrigeration.

PRINCIPLES OF REFRIGERATION

Nearly all refrigeration systems are based on the principle that heat is absorbed during evaporation of a liquid. The device which gives up heat to the liquid is thus cooled. The change in state of a substance from liquid to gas or vice versa varies with temperature and pressure. For example, water boils at 212° F at normal atmospheric pressure but boils at a lower temperature under reduced pressures. Similarly, the boiling point of refrigerant 12 varies with pressure. Figure 115 shows a simple refrigeration scheme whose main parts are: cooling unit or evaporator, compressor, and condenser.

The evaporator and condenser are partly filled with liquid refrigerant and the remaining parts with vapor. The compressor is a pump that runs continuously and removes vapors from the top of the evaporator and pumps them into the condenser. The low pressure in the evaporator and heat absorption cause the liquid to boil which causes the unit to cool through evaporation. The compressor action increases the pressure and the temperature of the vapor. The hot vapor is condensed into a liquid by transferring heat to cold water running through coils in the condenser.

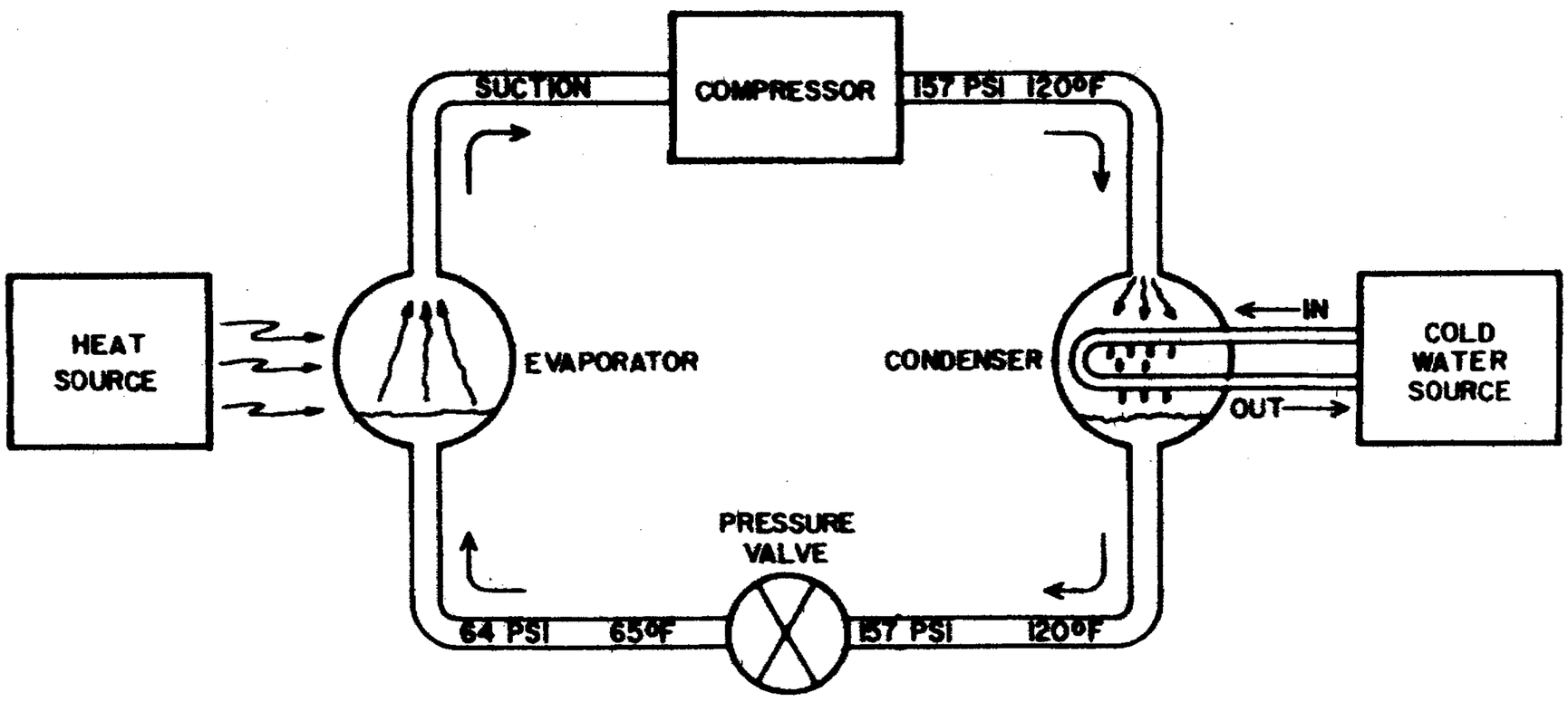


Figure 115. Simple Refrigeration Scheme

A pressure valve in the condenser to evaporator line regulates the pressure at which the liquid is returned to the evaporator. The pressure adjustment relates to the boiling point in the evaporator and hence to the temperature to be maintained. The liquid boils to cool the evaporator when the heat load raises the temperature of the liquid above its boiling point as determined by the pressure. The compressor pumps the vapors from the boiling liquid into the condenser (and increases the pressure) where they are converted to the liquid state and returned to the evaporator under controlled pressure and temperature. Thus the evaporator and an area surrounding it are maintained at a constant temperature.

SYSTEM DESCRIPTION

The cooling system contains the principal elements (evaporator, compressor, condenser) of common refrigeration systems plus additional regulating valves and gauges which are used for visual monitoring of performance. The refrigerant is refrigerant 12. Table 19 lists its characteristics for saturated conditions. The condenser is water cooled.

Table 19. Refrigerant 12 Characteristics

Temperature In Degrees F	Gauge Pressure
20	21 PSI
25	24.6
30	28.5
40	37
50	46.7
55	52
60	57.7
65	63.7
70	70.1
75	76.9
80	84.1
85	91.7
90	99.7
100	117.1
110	136.4
120	157.6
130	181

PHYSICAL DESCRIPTION

Figure 116 shows the various parts of the system. The evaporator and expansion valve are on each chassis (figure 116 and 117). Four chassis evaporators are connected in parallel to a common compressor, condenser, and related parts which are mounted in a frame assembly in the end of a wing. Each wing is a separate refrigeration system for four chassis with each chassis regulating its cooling with the varying logic and storage module heat load.

A control panel in each wing has separate temperature-pressure gauges for each chassis.

The evaporator coil is a copper tube imbedded in the aluminum bar that separates each row of modules in a chassis. Each module is secured to the aluminum bar and is separated from adjacent modules by black-finished metal spacers termed heat-sink plates. Each module fits in a separate compartment. Component heat from the modules flows by conduction and convection to the aluminum bar via the heat-sink plates and the metal module cap.

OPERATION

Liquid refrigerant from the condenser enters the top of the chassis through an expansion valve. The expansion valve causes a sharp pressure drop and controls the refrigerant flow by reading the outgoing refrigerant temperature and pressure. The chassis refrigerant temperature is held at 60° F by means of a pressure regulator valve on the chassis outlet. Heat from the aluminum bar raises the refrigerant temperature above the boiling point which produces evaporation and then cooling of the aluminum bars. Each bar is termed a cold bar and is cool to the touch. The aluminum face plate on the storage module and the logic module, the thin heat-sink plates between logic modules, the thick plates between storage modules, the air between modules and plates, the soldered component leads, and the printed circuit boards all form part of the heat conducting path from the discreet electronic component to the cold bar and then to the refrigerant.

The gaseous refrigerant returns to the compressor via the chassis pressure regulator valve at the bottom of the chassis (valve set to 57 psi) and a heat exchanger. The latter improves system efficiency and insures that no liquid refrigerant returns to the compressor during operation. The heat exchanger receives its heat from the hot liquid refrigerant produced by condensing the hot vapors from the high pressure (157 psi, 120° F) output of the compressor. The cold suction line which feeds the compressor passes through the heat exchanger jacket warming the refrigerant vapor as it passes to the compressor and cooling the liquid refrigerant.

ELEMENT DESCRIPTIONS

Accumulator

The accumulator prevents the compressor from slugging when certain room conditions cause liquid refrigerant to collect behind the suction solenoid valve during a long shut-down period. The unit has a built-in oil pick-up tube.

Compressor

The compressor is a high temperature, water cooled, semi-hermetic unit. It has a 208/230 volt, 60 cycle, 3-phase motor. The unit has its own current and temperature protection devices.

Condenser

The condenser is a water cooled unit for 40° F to 90° F (maximum) water. It is piped for cooling tower operation, but it is satisfactory for well operation. The unit is equipped with a 275 psi pressure relief valve or 162° F fusible plug.

Dual Pressure Control

The pressure control automatically shuts down the compressor by opening the starter if the compressor discharge pressure rises above the set point or if the suction pressure falls below the set point. The high pressure cut-out is normally set at 190 psi (fixed 40 psi differential), and the low pressure is normally set at 0 psi with a cut-in of 15 psi.

Evaporator Pressure Regulator

Regulator valves keep a constant set pressure (normally 60° F) on the chassis evaporator they control. The chassis pressure/temperature gauges are connected to the valve.

Expansion Valve

This valve controls the amount of refrigerant into its chassis evaporator, regardless of the load on the chassis, by means of a thermostatic element attached to the bottom evaporator tube. Normal superheat adjustment is from 4° to 9° F.

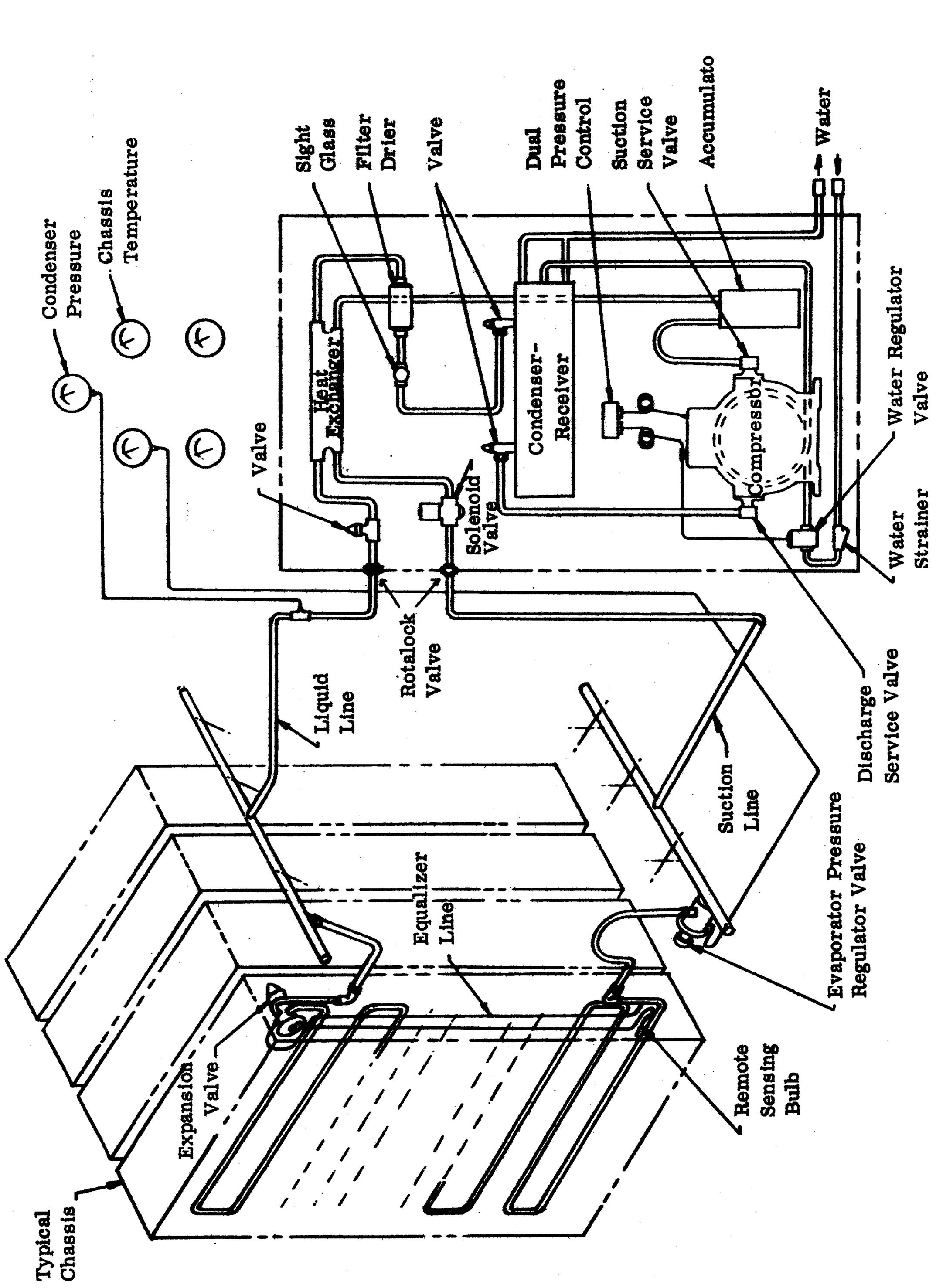


Figure 116. Control Data 6600 Conventional Cooling System

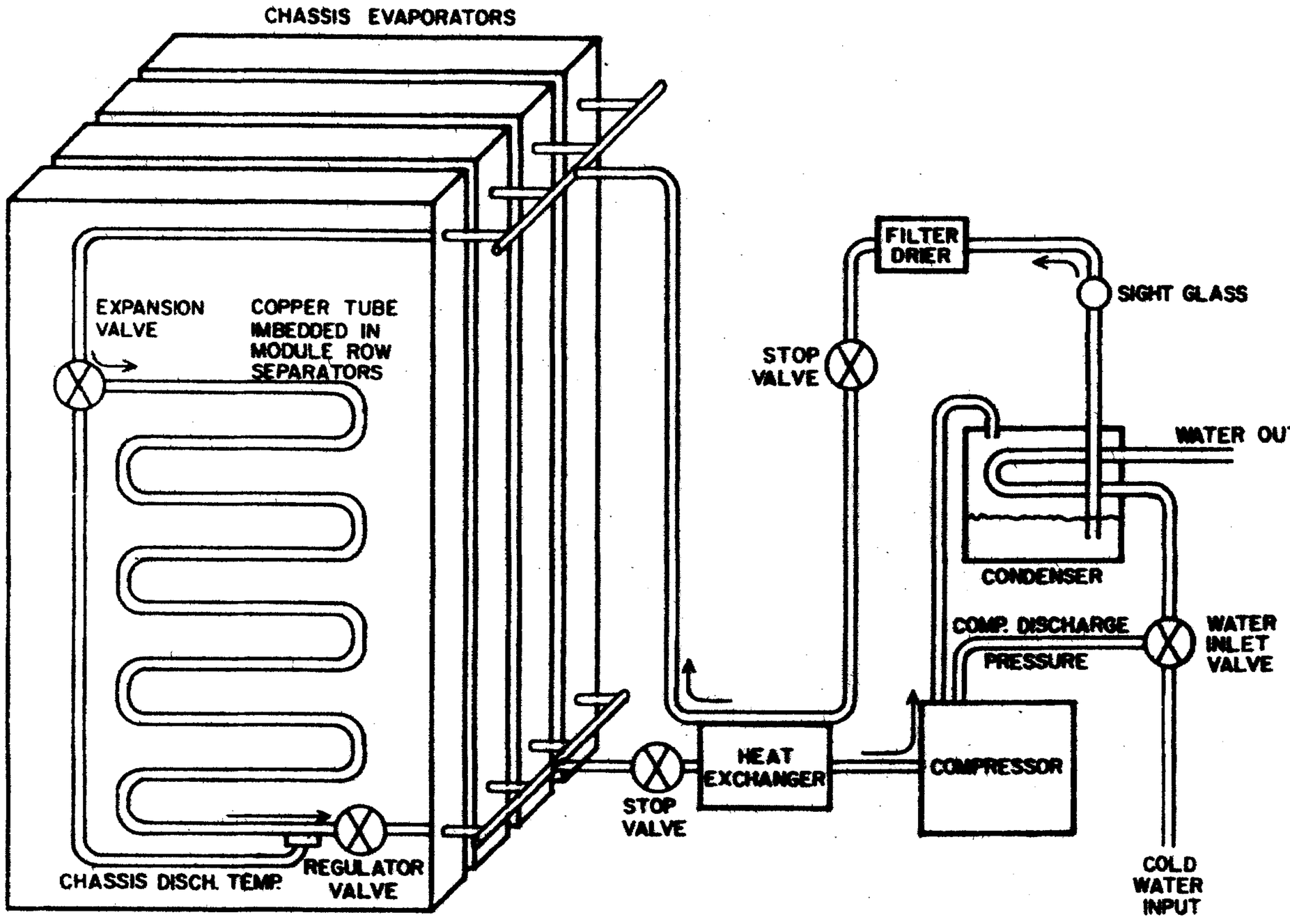


Figure 117. Simplified Diagram of 6000 Cooling System

Filter Drier

The drier removes moisture from the refrigerant, which enters the system by leaks, by opening the system for certain service operations or by a less than adequate evacuation process. It also removes foreign particles introduced during the assembly process.

Rotalock Valves

These are valves of the cabinet liquid and suction lines. They are used to keep the cabinet and chassis refrigerant lines from being opened to the atmosphere if a condensing unit must be replaced.

Sight Glass

The sight glass gives a visual check on whether there is sufficient refrigerant in the system and also whether or not the system is dry. If the dot is green, the system is dry; if the dot is yellow, the system is wet.

Solenoid Valve

This valve is closed during shut-down to prevent the refrigerant from migrating to the compressor. The solenoid is 115 volt, 60 cycle, and it is connected to the incoming power side of the compressor starter. The unit is equipped with a manual operating valve stem. It holds pressure against the chassis side only (not the condensing unit side).

Starter

This is a 3-phase compressor starter with 220 volt, 60 cycle coil. The unit does not have a protection device for the compressor.

Water Regulating Valve

This is an automatically regulating valve used to keep the condensing temperature at a set value (normally 120° F). It also closes off water flow during unit shut-down. The valve is operated off the compressor head pressure.

WARNING SYSTEM

A warning system is built into each wing of the 6600 system to protect the modules against high temperature and condensation if a cooling system malfunction or adverse room air conditions occur. If a

malfunction occurs, a relay activates a warning buzzer and indicator light. After a short period of time (approximately two minutes) the 400 cycle and 60 cycle power is automatically turned off.

If the gauge pressure for any one chassis drops below the minimum setting of the red hand (57° F), the Gauge switch starts the sequence for operating the warning buzzer, the low temperature indicator for that chassis, and the time delay relay for shutting down the power. A low gauge pressure indicates a low temperature condition which can cause condensation on the chassis (see K on the troubleshooting chart).

If the gauge pressure on any one chassis rises above the maximum setting of the red hand or if the thermostat on the bottom cold bar closes (at approximately 85° F), the buzzer (the high temperature indicator for that chassis) and the time delay relay are turned on (see J on the troubleshooting chart).

If trouble is indicated it is very important that the settings on the gauges or time delay relays not be altered, because alteration could result in serious damage to the modules in the affected chassis.

When trouble is indicated, the customer engineer should immediately check the refrigeration maintenance panel to find out on which chassis the cooling has failed, whether the high or low temperature light is on, and whether the gauge for the chassis is the activating source. If more than one chassis temperature indicator is lit, check the condenser gauge (it should be approximately 120°) and the compressor indicator light to see if the compressor is running or short cycling. If this information can be obtained before the machine shuts down, the source of trouble can probably be tracked down by using the troubleshooting chart found at the end of this chapter.

Another part of the warning system, but not an inherent part of the computer, is a chart recorder assembly for the room temperature and dew point. This instrument is used to keep a daily record of the room conditions. It also has an audible (buzzer) warning device if the dew point goes above 50° F.

PREVENTIVE MAINTENANCE

DAILY

Whenever the computer is started, listen for short cycling of any of the compressors.

Check the room conditions for the dew point and temperature. Do not run the computer if the dew point goes over the maximum of 50° F dew point. Serious computer damage can result if the room air dew point rises to the chassis temperature.

After the machine has run for approximately 10 minutes, check the condenser pressure and all chassis gauges on each maintenance panel. The condenser pressure should read approximately 157 psig or 120° F and steady. All the chassis gauges should be indicating well within the limit hands. They should read 60° to 62° F with the limit hands set at 57° F and 72° F.

WEEKLY

In addition to performing the daily routine, the following procedures should be performed once a week. Check compressor oil level before startup and after the computer has been running for an hour. If the oil level lowers to the bottom of the glass immediately after startup, it probably indicates one or more chassis are flooding (see C-3 and G on the troubleshooting chart). The glass should always be between one quarter and three quarters full.

The liquid line sight glass should be checked for bubbles after the system has been operating for 30 minutes. If bubbles are present or the system is not running full, see B-5 on the troubleshooting chart. Also check the color of the dot in the center of the sight glass. If the dot is yellow-tinged (not green), moisture has entered the system beyond the capacity of the drier. The leak must be found, repaired, and a new filter drier must be installed.

CAUTION

The compressor will burn out if the system is allowed to operate very long in this condition.

Unless rust-inhibited and filtered water is used, the water should be drained from the condensing system and a careful inspection made of the condenser tubes by removing the condenser end plate (s). The tubes should be cleaned and the scale removed. The water pipe strainer on each condensing unit also should be cleaned. Check the condition of the contacts in the pressure control and starter on each condensing unit.

GENERAL REFRIGERATION SERVICE

PUMP-DOWN PROCEDURE

- Attach pressure gauge to compressor suction valve port and crack the valve off the back seat so the gauge will function.
- Close the liquid line valve on the condenser.
- Open the evaporator pressure regulator valves to minimum setting (40 psi). This step is not necessary unless the liquid lines on the chassis end of the system are to be opened.

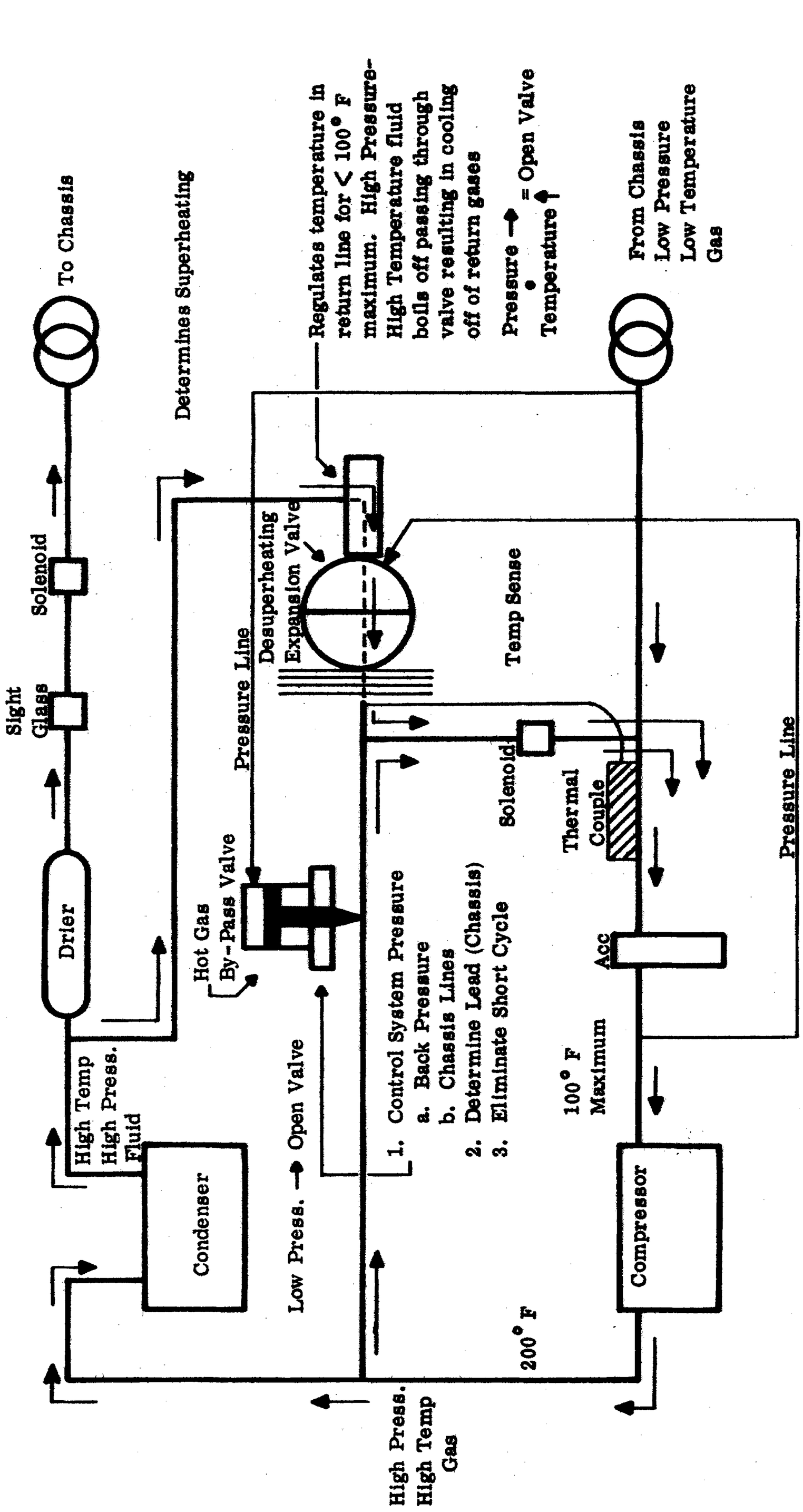


Figure 118. 6000 Refrigeration System (New Style)

- Start the compressor and allow it to run until the suction gauge indicates two psig. At this point, stop the compressor and quickly close the compressor discharge line. Also close the valve on the condenser inlet. Leave the small pressure remaining in the lines so that air and moisture will not enter when the system is opened.

NOTE

If the refrigerant lines to be opened are colder than the ambient dew point temperature, sweating will occur on the inside as well as the outside surface. Allow the piping to warm up to room temperature before opening it up to the air.

FILTER-DRIER REPLACEMENT

If a filter drier needs replacing, pump down the condensing unit in accordance with the preceding description. However, it is not necessary to open the pressure regulator valves. After the unit has been pumped down, close the valve in the liquid line, loosen the drier on the unit so it can be removed quickly, remove the seals on the new drier, and, as quickly as possible, remove the old drier and install the new one. Put the system back into operation and leak-check all joints around the drier.

ADDING REFRIGERANT

Refrigerant should be added through the compressor suction valve port. To add refrigerant, complete the following steps.

- Connect the refrigerant drum to the valve port with a charging line. Purge air from the line before tightening the hose connection onto the compressor valve.
- Hold the refrigerant drum in a vertical position with the charging line at the top so that only gas can enter the system.
- The compressor can be operated in the normal manner while charging. Crack the suction valve off the back seat and open the refrigerant drum valve so gas can enter the system.
- Observe the sight glass. When the charge is adequate, no bubbles can be seen and the bottom of the condenser is cool. Shut off the drum valve and close the compressor valve to the back seat.

  Check the compressor oil level.

PURGING

If air or other noncondensable gases are present in the system, they tend to collect in the condenser. Such gases can enter when part of the system has been opened for servicing which occurs during drier replacement or addition of oil. When noncondensable gases are present, the head pressure rises to a point above the pressure corresponding to the temperature at which the vapor is condensing.

Before the need to purge the system can be determined, the entire system must stand idle long enough to cool down to room temperature. This is usually done overnight with the condensing unit door open and the panels off. When the system has attained room temperature, the condenser gauge on the panel should not read more than 10 psi above the saturation pressure at room temperature.

To purge pump down the system and close the compressor discharge valve after stopping the compressor. Allow the condenser water to run for approximately five minutes to condense as much vapor as possible. The water regulating valve must be completely open for the water to run. After the water has run for five minutes, purge through the compressor discharge port by opening the valve for approximately five seconds. Repeat the purging step several times at about three minute intervals with the condenser water running continuously. After purging the system can be placed back into service.

CONDENSING UNIT REPLACEMENT

Complete the following steps to replace the condensing unit.

- Use the recommended pump-down procedure.
- After the unit has been pumped down, close the following valves: compressor suction, cabinet suction, cabinet liquid, and the condensing unit liquid line valve.
- On serial number 1 only, disconnect the condenser gauge line and insert plug.
- Shut down the Main Power switch and disconnect the power cable between the cabinet distribution box and the condensing unit.
- Remove condensing unit hold-down bolts in the bottom of the frame.
- Disconnect the liquid and suction lines at the cabinet shut-off valves.
- Slide the condensing unit out. Be careful not to hook any tubing on either the cabinet or the condensing unit.
- Slide in the replacement unit after removing the liquid and suction line caps. Place caps on the old unit.
- Reverse the first six steps.

- Place unit into operation, leak check, observe sight glass for sufficient charge, and check compressor oil level.

CHASSIS PRESSURE REGULATOR VALVE SERVICING

If this valve will not hold its set pressure it might be obstructed (resulting in a high gauge pressure), stuck open (resulting in a low gauge pressure), or oil may have collected in the chassis (resulting in a low gauge pressure) due to a very light load on that chassis. If one of the first two problems is present, it might be remedied by opening the valve to the maximum and then resetting it. A light load cannot be permanently remedied without adding load to that chassis.

If opening and resetting the valve does not appear to help, the gauge, gauge line, and actual chassis temperature should be checked before resorting to valve replacement.

First, check the actual chassis temperature against the indicated gauge temperature. Do this on any cold bar except the bottom one. Temperature of the bar should be not more than 5° F above the indicated gauge temperature. If there is a wide disparity, the gauge or gauge line is the problem, or the chassis is loaded with oil resulting in a much higher actual temperature than the gauge indicates.

Check the gauge line out by loosening the line at the gauge enough to see if a flow of refrigerant is obtained. If the line is not clogged, shut off the gauge line valve on the regulator by removing the cap on the underside. Then close the gauge line valve with a screwdriver. Attach a service gauge in place of the panel gauge, open the gauge line valve, and observe the pressures obtained while the system is in operation. If satisfactory readings are obtained, reconnect the line to the panel gauge and recheck. If panel gauge readings are still erroneous, replace the gauge.

If the gauge line seems to be clogged, it may be possible to blow it clear by disconnecting the line from the gauge and operating the system. If this is done, the serviceman must be ready to close off the gauge line by the gauge valve very quickly or oil will be blown all over the unit. If this procedure does not clear the line, both ends of the line must be disconnected and dry nitrogen forced through the line.

If valve replacement is required, complete the following procedures.

- Open the pressure regulator valve, which must be removed, to the minimum setting.
- Pump down the system and close off the cabinet liquid and suction lines.
- Remove the suction manifold cover, gauge line, hose connector, and the valve connector.
- Replace the old valve with new one and make all connections. Open the gauge line valve.
- Open all valves and start system.
- Set valve to 55° through 57° F, leak check, and check sight glass and oil sump.

EXPANSION VALVE SERVICE

The four following descriptions give procedures for expansion valve servicing.

Superheat Setting

If it is suspected that the superheat setting is incorrect (see C-3, D-4, G-1, and I-1 on the troubleshooting chart), it can be easily checked by getting the temperature of the chassis lower hose fitting (place thermocouple on the plated tube of the fitting) and getting the temperature difference between it and the indicated gauge temperature. The actual temperature should be 4° to 8° F above the indicated gauge temperature. Because the chassis evaporator is quite high, the temperature at the hose fitting may not be constant. It is wise to take several readings over a period of a few minutes.

If the superheat is incorrect (a large variation of 10° or 15° F indicates too high a superheat setting), remove the cap on the back of the valve (wired side of the chassis). Be sure to place a wrench on the valve body to keep it from twisting. Turn in (clockwise) the valve stem to increase superheat; turn out to decrease superheat. When set, replace valve cap snugly.

Power (Thermostat) Element

The power element of the expansion valve consists of a remote bulb, a capillary tube, and the diaphragm. If this power element has lost its charge, the valve will maintain a closed, or nearly closed, position (see H-3, and D-3 on the troubleshooting chart). If this is suspected, due to a high chassis temperature which is not gauge indicated or because of a very high superheat which is not adjustable, check the power element by:

- Removing the bottom transformer cover and removing the bulb from the tube on the chassis
- Holding the bulb in your hand and listening at the valve with a stethoscope or by measuring the temperature at the lower hose connection, whether the valve opens up or not

If the temperature does not drop down to a reasonable superheat, replace the power element using the procedure listed below.

- Pump down the system after screwing out all the valves.
- Remove the top transformer cover, the hinge side cover, and remove the bulb capillary from its clamps.
- Unscrew the valve diaphragm. Hold the valve body with a wrench to keep it from turning.
- Replace the old power element with the new element. Coil the capillary and clamp as before. Clamp the bulb to the top of the tube and insulate. The capillary coil must be coiled and taped so it does not rub itself or other parts. The capillary must also be drawn tightly through the clamps.
- Start up the system, reset the pressure regulator valve, and leak check around the diaphragm.
- Check and set the superheat setting. Replace the hardware.

Inner Valve Parts Inspection and Servicing

If an expansion valve indicates that it is obstructed (see D-3 on the troubleshooting chart), follow the procedure listed below.

- Pump down the system with all the pressure regulator valves opened up.
- Slowly remove the bottom cap assembly as the pressure is relieved. Hold the valve body when loosening the bottom cap to keep the valve from twisting.
- Inspect the inner parts, especially the valve seat, for foreign matter. Clean them out with a lintless rag.
- Inspect the valve seat and pin for wear or cuts. Replace it if necessary. If the seat has to be replaced, the diaphragm must be removed in order to get the push rods out of the way.
- Reassemble the valve, start the system, and reset the pressure regulator valves.
- Leak check the valve and adjust the superheat.
- Check the refrigerant charge.

External Equalizer Line

If the external equalizer becomes plugged or restricted (see C-3 and G-1 on the troubleshooting chart), the expansion valve will feed too much which can result in a low superheat condition (possibly not adjustable) or none at all, which can cause flood-back to the compressor. If this condition is suspected, it can be checked out and possibly remedied by following the procedure listed below.

- Pump down the system after opening the affected chassis pressure regulator valve. Close off the cabinet liquid and suction lines.
- Remove the equalizer line at the expansion valve and cap the valve connection.
- Crack the liquid line valves momentarily to allow the refrigerant to flow through the chassis and blow out the equalizer line.
- If sufficient pressure cannot be applied in this manner, attach dry nitrogen to the equalizer line and apply 250 psi. Do so in short timed spurts because the nitrogen must be purged off later.
- Reconnect the line, start the system, and reset the pressure regulator.
- Check the superheat setting, leaks, and observe the sight glass.

Table 20. Troubleshooting Chart for 6600 Cooling System

COMPLAINT	POSSIBLE CAUSE	SYMPTOMS	RECOMMENDED ACTION
A. Compressor does not start	1. Open breaker	1. No current to line side of contactor	1. Determine why breaker was open, whether due to manual trip or overload on compressor.
	2. Burned-out motor	2. Full voltage on motor terminals but motor does not run	2. Replace condensing unit assembly. Use standard cleaning procedures for chassis evaporators if required.
	3. Open motor protector	3. Full voltage to motor protector but not to motor terminals	3. Replace protector.
	4. Dual pressure control a. Faulty control b. Suction pressure cut-in setting c. Discharge pressure cut-in setting	4. Motor contactor coil not energized a. Pressures correct for settings but contact remains open b. Suction pressure below cut-in setting c. Discharge pressure above cut-in setting	4. a. Repair or replace. b. Check for loss of refrigerant. Repair leak or replace condensing unit. See complaint H. c. See complaint E.
B. Compressor short cycles	1. Low pressure control, differential set too close	1. Normal operation except frequent stops and starts	1. Set differential to job conditions.
	2. High pressure control, cut-out set too close	2. Normal operation except frequent stops and starts	2. Set to job conditions.
	3. Faulty condensing	3. Excessively high discharge pressure	3. Check for water failure. See complaint E.
	4. Overcharge of refrigerant or noncondensable gas	4. High discharge pressure	4. Remove excess refrigerant or purge noncondensables.
	5. Low refrigerant charge	5. Normal operation except frequent stops and starts; sight glass partially full or contains bubbles	5. Repair refrigerant leak and recharge.
	6. Overcharge of compressor oil	6. Low suction and discharge pressure	6. Remove excess oil.
	7. Water regulating valve inoperative or restricted or water temperature is too high	7. High discharge pressure	7. Clean or repair valve or correct temperature of water.
	8. Water piping restricted or supply pressure is too low	8. High discharge pressure	8. Correct cause.
	9. Restricted drier strainer	9. Low suction pressure and frosting at strainer	9. Replace strainer.
	10. Fouled condenser	10. High discharge pressure, cool outlet water	10. Clean condenser tubes.

Table 20. Troubleshooting Chart for 6600 Cooling System (Cont)

COMPLAINT	POSSIBLE CAUSE	SYMPTOMS	RECOMMENDED ACTION
C. Compressor is noisy	1. Lack of oil	1. Oil level cannot be seen in glass	1. Add oil.
	2. Internal compressor parts broken	2. Compressor knocks	2. Replace condensing unit assembly.
	3. Liquid flood-back	3. Abnormally code suction line; compressor knocks	3. Check expansion valve superheats. Check expansion valve bulbs for looseness. Check chassis gauges for low pressure. Check for restricted expansion valve external equalizer.
	4. Dirty water regulating valve; too high water pressure or intermittent water pressure	4. Water valve hammers or chatters	4. Clean water regulating valve. Install air chamber ahead of valve.
D. System short of capacity (one or more chassis evaporators running too warm)	1. Flash gas in liquid line	1. Expansion valve hisses; shows in sight glass	1. Add refrigerant or lower condensing temperature.
	2. Clogged drier strainer	2. Temperature drop through strainer	2. Replace strainer.
	3. Expansion valve stuck or obstructed or bad power element	3. Chassis gauge pressure correct but chassis is several degrees warmer	3. Clean or repair expansion valve.
	4. Improper superheat adjustment	4. Chassis gauge pressure correct but bottom part of chassis is several degrees warmer than top part	4. Check expansion valve bulb for looseness. Reset superheat setting of valve.
	5. Worn compressor valves	5. High suction pressure nearly equal to discharge	5. Install new compressor valves and valve plate.
	6. Chassis pressure regulator valve stuck or obstructed	6. Chassis gauge pressure too high and chassis too warm	6. If valve won't regulate, replace valve.
E. Condenser pressure too high	1. Too little or too warm condenser water	1. Excessively warm water leaving condenser	1. Provide adequate cool water. Adjust water valve.
	2. Fouled tubes in condenser	2. Excessively cool water leaving condenser	2. Clean tubes.
	3. Air or noncondensable gas in system	3. Exceptionally hot condenser and excessive discharge pressure	3. Purge.

Table 20. Troubleshooting Chart for 6600 Cooling System (Cont)

COMPLAINT	POSSIBLE CAUSE	SYMPTOMS	RECOMMENDED ACTION
E. Condenser pressure too high (Cont)	4. Overcharge of refrigerant	4. Excessive discharge pressure, condition excessively hot (bottom third or less should be cool)	4. Purge.
F. Condenser pressure too low	1. Too much condenser water	1. Cold water leaving condenser	1. Adjust water regulating valve.
	2. Lack of refrigerant	2. Bubbles in sight glass	2. Repair leak and charge or replace condensing unit.
	3. Broken or leaking compressor discharge valve	3. Suction pressure rises faster than five pounds per minute after shutdown (can also happen with leaking suction solenoid valve)	3. Repair compressor or replace condensing unit.
G. Suction pressure too high	1. Overfeeding of expansion valve	1. Abnormally cold suction line or liquid flood-back	1. Check expansion valve remote bulb for being properly attached. Adjust superheat setting. Check for restricted expansion valve external equalizer.
	2. Expansion valve stuck in open position	2. Abnormally cold suction line or liquid flood-back	2. Repair or replace valve.
	3. Pressure regulator valve stuck in open position	3. Abnormally cold suction line or liquid flood-back or low chassis gauge pressure	3. Tap valve lightly or replace.
	4. Broken or leaking valves in compressor	4. Noisy compressor or rapid pressure change after shutdown	4. Replace valves on compressor or replace condensing unit assembly.
H. Suction pressure too low	1. Lack of refrigerant	1. Bubbles in sight glass	1. Repair leak or replace condensing unit assembly.
	2. Clogged drier-strainer	2. Temperature change of refrigerant line through strainer	2. Replace drier-strainer.
	3. Loss of charge in expansion valve power assembly	3. No refrigerant flow through valve; chassis excessively warm	3. Replace expansion valve power assembly.
	4. Obstructed expansion valve	4. Loss of capacity; chassis is excessively warm	4. Clean or repair valve.
	5. Obstructed chassis pressure regulator valve	5. Loss of capacity; chassis gauge pressure is high	5. Replace valve.

Table 20. Troubleshooting Chart for 6600 Cooling System (Cont)

COMPLAINT	POSSIBLE CAUSE	SYMPTOMS	RECOMMENDED ACTION
I. Compressor knock on start-up and frosting suction manifold	1. Expansion valve stuck in open position	1. Gauge pressure correct, frost on suction line, or rattling knock of compressor on start-up	1. Check expansion valves for superheat. Clean or replace internal parts.
J. Buzzer and high temperature light for one chassis	1. Expansion valve obstructed or stuck in closed position	1. Gauge pressure correct but chassis has abnormally high temperature (high temperature thermostat is controlling)	1. Clean out or replace internal parts of expansion valve.
	2. Pressure regulator valve obstructed or stuck in closed position	2. Chassis gauge pressure abnormally high and chassis temperature abnormally high (gauge oil thermostat is controlling)	2. Open valve to maximum and reset or replace valve.
K. Buzzer and low temperature light for one chassis	1. Pressure regulator valve stuck in open position	1. Chassis gauge pressure abnormally low and chassis temperature abnormally cold (gauge is controlling)	1. Open and close valve to maximum settings and reset or replace valve.
L. Noisy pressure regulator valve	1. Adjustment spring	1. Noisy valve(s) with noticeable vibrations	1. Remove adjusting cap; lightly grease spring and brass washer (end-for-end spring).

APPENDIX A
PERIPHERAL PROCESSOR COMMAND TIMING

Table A-1. Peripheral and Control Processor Instructions

Mnemonic and Octal Code	Name	Mnemonic and Octal Code	Name
PSN 00	Pass	LDI 40	Load ((d))
LJM 01	Long Jump to m + (d)	ADI 41	Add ((d))
RJM 02	Return Jump to m + (d)	SBI 42	Subtract ((d))
UJN 03	Unconditional Jump d	LMI 43	Logical Difference ((d))
ZJN 04	Zero Jump d	STI 44	Store ((d))
NJN 05	Nonzero Jump d	RAI 45	Replace Add ((d))
PJN 06	Plus Jump d	AOI 46	Replace Add One ((d))
MJN 07	Minus Jump d	SOI 47	Replace Subtract One ((d))
SHN 10	Shift d	LDM 50	Load (m + (d))
LMN 11	Logical Difference d	ADM 51	Add (m + (d))
LPN 12	Logical Product d	SBM 52	Subtract (m + (d))
SCN 13	Selective Clear d	LMM 53	Logical Difference (m + (d))
LDN 14	Load d	STM 54	Store (m + (d))
LCN 15	Load Complement d	RAM 55	Replace Add (m + (d))
ADN 16	Add d	AOM 56	Replace Add One (m + (d))
SBN 17	Subtract d	SOM 57	Replace Subtract One (m + (d))
LDC 20	Load dm	CRD 60	Central Read from (A) to d
ADC 21	Add dm	CRM 61	Central Read (d) Words from (A) to m
LPC 22	Logical Product dm	CWD 62	Central Write to (A) from d
LMC 23	Logical Difference dm	CWM 63	Central Write (d) Words to (A) from m
PSN 24	Pass	AJM 64	Jump to m if Channel d Active
PSN 25	Pass	IJM 65	Jump to m if Channel d Inactive
EXN 26	Exchange Jump	FJM 66	Jump to m if Channel d Full
RPN 27	Read Program Address	EJM 67	Jump to m if Channel d Empty
LDD 30	Load (d)	IAN 70	Input to A from Channel d
ADD 31	Add (d)	IAM 71	Input (A) Words to m from Channel d
SBD 32	Subtract (d)	OAN 72	Output from A on Channel d
LMD 33	Logical Difference (d)	OAM 73	Output (A) Words from m on Channel d
STD 34	Store (d)	ACN 74	Activate Channel d
RAD 35	Replace Add (d)	DCN 75	Disconnect Channel d
AOD 36	Replace Add One (d)	FAN 76	Function (A) on Channel d
SOD 37	Replace Subtract One (d)	FNC 77	Function m on Channel d

PERIPHERAL COMMAND TIMING

(00) Read P; P+1 to P.

(01) Read P; P+1 to P; d to Q; F to K; Advance K if d=0.
010 Read Q; Fd to Q; Advance K.
011 Read P; Q+Fd to P; Clear K.

(02) Read P; P+1 to P; d to Q; F to K; Advance K if d=0.
020 Read Q; Fd to Q; Advance K.
021 Read P; Q+Fd to Q; P+1 to P; Advance K.
022 Read Q; P to Z; Q+1 to P; Clear K.

(03) Read P; P+d to P.

(04) Read P; P+d to P if A=0; P+1 to P if A≠0.

(05) Read P; P+d to P if A≠0; P+1 to P if A=0.

(06) Read P; P+d to P if A is +; P+1 to P if A is -.

(07) Read P; P+d to P if A is -.

(10) Read P; P+1 to P; Shift A by d.

(11) Read P; A ld to A; P+1 to P.

(12) Read P; A lp d to A; P+1 to P.

(13) Read P; A sc d to A; P+1 to P.

(14) Read P; d to A; P+1 to P.

(15) Read P; -d to A; P+1 to P.

(16) Read P; A+d to A; P+1 to P.

(17) Read P; A-d to A; P+1 to P.

(20) Read P; d to Q; P+1 to P; F to K.
200 Read P; QFd to A; P+1 to P; Clear K.

(21) Read P; d to Q; P+1 to P; F to K.
210 Read P; A+QFd to A; P+1 to P; Clear K.

(22) Read P; d to Q; P+1 to P; F to K.
220 Read P; A lp QFd to A; P+1 to P; Clear K.

(23) Read P; d to Q; P+1 to P; F to K.
230 Read P; A ld QFd to A; P+1 to P; Clear K.

(24) Read P; P+1 to P.

(25) Read P; P+1 to P.

(26) Read P; A to L; If central empty (P+1 to P).

(27) Read P; d=00X0 A; d=01X1-A; P+1 to P.

(30) Read P; d to Q; P+1 to P; F to K.
300 Read Q; Fd to A; Clear K.

(31) Read P; d to Q; P+1 to P; F to K.
310 Read Q; A+Fd to A; Clear K.

(32) Read P; d to Q; P+1 to P; F to K.
320 Read Q; A-Fd to A; Clear K.

(33) Read P; d to Q; P+1 to P; F to K.
330 Read Q; A Ld Fd to A; Clear K.

(34) Read P; d to Q; P+1 to P; F to K.
340 Read Q; A to Z; Clear K.

(35) Read P; d to Q; P+1 to P; F to K.
350 Read Q; A+Fd to A; Set K=340.

(36) Read P; d to Q; P+1 to P; F to K.
360 Read Q; Fd+1 to A; Set K=340.

(37) Read P; d to Q; P+1 to P; F to K.
370 Read Q; Fd-1 to A; Set K=340.

(40) Read P; d to Q; P+1 to P; F to K.
400 Read Q; Fd to Q; Advance K.
401 Read Q; Fd to A; Clear K.

(41) Read P; d to Q; P+1 to P; F to K.
410 Read Q; Fd to Q; Advance K.
411 Read Q; A+Fd to A; Clear K.

(42) Read P; d to Q; P+1 to P; F to K.
420 Read Q; Fd to Q; Advance K.
421 Read Q; A-Fd to A; Clear K.

(43) Read P; d to Q; P+1 to P; F to K.
430 Read Q; Fd to Q; Advance K.
431 Read Q; A Ld Fd to A; Clear K.

(44) Read P; d to Q; P+1 to P; F to K.
440 Read Q; Fd to Q; Advance K.
441 Read Q; A to Z; Clear K.

(45) Read P; d to Q; P+1 to P; F to K.
450 Read Q; Fd to Q; Advance K.
451 Read Q; A+Fd to A; Set K=340.

(46) Read P; d to Q; P+1 to P; F to K.
460 Read Q; Fd to Q; Advance K.
461 Read Q; Fd+1 to A; Set K=340.

(47) Read P; d to Q; P+1 to P; F to K.
470 Read Q; Fd to Q; Advance K.
471 Read Q; Fd-1 to A; Set K=340.

(50) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.
500 Read Q; Fd to Q; Advance K.
501 Read P; Q+Fd to Q; P+1 to P; Advance K.
502 Read Q; Fd to A; Clear K.

(51) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.
510 Read Q; Fd to Q; Advance K.
511 Read P; Q+Fd to Q; P+1 to P; Advance K.
512 Read Q; A+Fd to A; Clear K.

(52) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.

520 Read Q; Fd to Q; Advance K.
521 Read P; Q + Fd to Q; P+1 to P; Advance K.
522 Read Q; A - Fd to A; Clear K.

(53) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.
530 Read Q; Fd to Q; Advance K.
531 Read P; Q + Fd to Q; P+1 to P; Advance K.
532 Read Q; A Ld to A; Clear K.

(54) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.
540 Read Q; Fd to Q; Advance K.
541 Read P; Q + Fd to Q; P+1 to P; Advance K.
542 Read Q; A to Z; Clear K.

(55) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.
550 Read Q; Fd to Q; Advance K.
551 Read P; Q + Fd to Q; P+1 to P; Advance K.
552 Read Q; A + Fd to A; Set K=340.

(56) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.
560 Read Q; Fd to Q; Advance K.
561 Read P; Q + Fd to Q; P+1 to P; Advance K.
562 Read Q; Fd+1 to A; Set K=340.

(57) Read P; d to Q; P+1 to P; F to K; Advance K if d=0.
570 Read Q; Fd to Q; Advance K.
571 Read P; Q + Fd to Q; P+1 to P; Advance K.
572 Read Q; Fd=1 to A; Set K=340.

(60) Read P; A to L; d to Q. If C1 + C2 + C3 + C4 Empty and C5 Empty and Central $\overline{\text{Busy}}$ (P+1 to P; F to K).
600 Read Q. If C5 Full and C4 Empty (C5 to Z; C5 to C4; Q+1 to Q; Advance K).
601 Read Q. If C3 Empty (C4 to Z; C4 to C3; Q+1 to Q; Advance K).
602 Read Q. If C2 Empty (C3 to Z; C3 to C2; Q+1 to Q; Advance K).
603 Read Q. If C1 Empty (C2 to Z; C2 to C1; Q+1 to Q; Advance K).
604 Read Q; C1 to Z; Clear K.

(61) Read P; d to Q; P+1 to P; F to K.
610 Read Q; Fd to Q; Advance K.
611 Read 0; P to Z; Advance K.
612 Read P; A to L. If C5 Empty and Central $\overline{\text{Busy}}$ and C1 or C2 or C3 or C4 Empty (Fd to P; Advance K).
613 Read P. If C5 Full and C4 Empty (C5 to Z; C5 to C4; P+1 to P; Advance K).
614 Read P. If C3 Empty (C4 to Z; C4 to C3; P+1 to P; A+1 to A; Advance K).
615 Read P. If C2 Empty (C3 to Z; C3 to C2; P+1 to P; Q-1 to Q; Advance K).
616 Read P. If C1 Empty (C2 to Z; C2 to C1; P+1 to P; Advance K).
617 Read P. If Q=0 (C1 to Z; K=713). If Q≠0 and Central $\overline{\text{Busy}}$ and C5 Empty (C1 to Z; A to L; P+1 to P; K=613).

(62) Read P; d to Q; P+1 to P; F to K.
620 Read Q. If D1 Empty (Fd to D1; Q+1 to Q; Advance K).
621 Read Q. If D2 Empty (Fd to D2; D1 to D2; Q+1 to Q; Advance K).
622 Read Q. If D3 Empty (Fd to D3; D2 to D3; Q+1 to Q; Advance K).
623 Read Q. If D4 Empty (Fd to D4; D3 to D4; Q+1 to Q; Advance K).
624 Read Q; A to L. If Central $\overline{\text{Busy}}$ (Fd to D5; D4 to D5; A to L; Clear K).

(63) Read P; d to Q; P+1 to P; F to K.
630 Read Q; Fd to Q; Advance K.
631 Read 0; P to Z; Advance K.
632 Read P; Fd to P; Advance K.
633 Read P. If D1 Empty (Fd to D1; P+1 to P; Advance K).
634 Read P. If D2 Empty (Fd to D2; D1 to D2; P+1 to P; Advance K).
635 Read P. If D3 Empty (Fd to D3; D2 to D3; P+1 to P; Q-1 to Q; Advance K).
636 Read P. If D4 Empty (Fd to D4; D3 to D4; P+1 to P; Advance K).
637 Read P; A to L. If Central $\overline{\text{Busy}}$ (Fd to D5; D4 to D5; A+1 to A; P+1 to P). If Central $\overline{\text{Busy}}$ and Q=0 (K=733). If Central $\overline{\text{Busy}}$ and Q≠0 (K=633).

(64) Read P; d to Q; P+1 to P; F to K.
640 Read P; Fd to P if Channel Active; P+1 to P if Channel Inactive; Clear K.

(65) Read P; d to Q; P+1 to P; F to K.
650 Read P; Fd to P if Channel Inactive; P+1 to P if Channel Active.

(66) Read P; d to Q; P+1 to P; F to K.
660 Read P; Fd to P if Channel Full; P+1 to P if Channel Empty; Clear K.

(67) Read P; d to Q; P+1 to P; F to K.
670 Read P; Fd to P if Channel Empty; P+1 to P if Channel Full; Clear K.

(70) Read P; d to q; P+1 to P; F to K.
700 Read P; R to A; If Channel Full (Clear K).

(71) Read P; d to Q; P+1 to P; F to K.
710 Read 0; P to Z; Advance K.
711 Read P; Fd to P; Advance K.
712 Read P. If Channel Disconnect (Advance K). If Channel Full (R to Z; P+1 to P; A-1 to A; Advance K if A=1).

713 Read 0; Fd+1 to P; Clear K.

(72) Read P; d to Q; P+1 to P; F to K.

720 Read P. If Channel Empty and Active (A to R; Clear K).

(73) Read P; d to Q; P+1 to P; F to K.

730 Read 0; P to Z; Advance K.

731 Read P; Fd to P; Advance K.

732 Read P. If Channel Disconnect (Advance K). If Channel Empty Active (Fd to R; P+1 to P; A-1 to A; Advance K if A=1).

733 Read 0; Fd+1 to P; Clear K.

(74) Read P; d to Q; P+1 to P; F to K.

740 Read P. If Channel Inactive (Activate Channel, Clear K).

(75) Read P; d to Q; P+1 to P; F to K.

750 Read P. If Channel Active (Disconnect Channel, Clear K).

(76) Read P; d to Q; P+1 to P; F to K.

760 Read P. If Channel Inactive (A to R; Function; Clear K).

(77) Read P; d to Q; P+1 to P; F to K.

770 Read P. If Channel Inactive (Fd to R; Function; P+1 to P; Clear K).

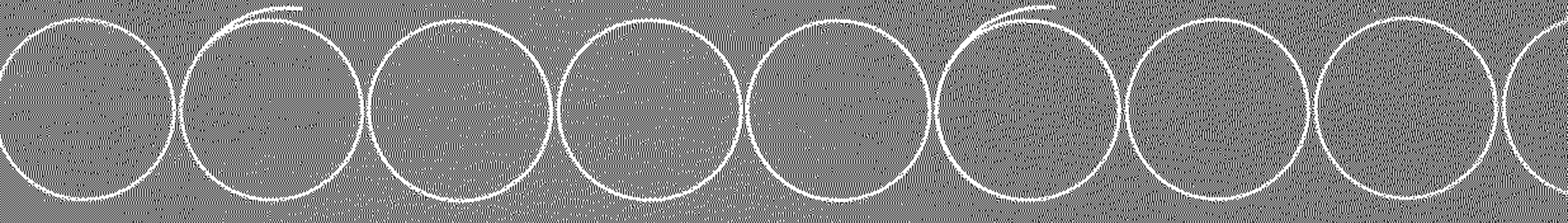
COMMENT SHEET

6000 Series Introduction
and Peripheral Processors Training Manual
First Edition

FROM: Name _____

Address _____

COMMENTS: Describe errors, suggested additions or deletions, and include page numbers with each reference.



CONTROL DATA INSTITUTES

Arlington, Virginia 22204
3217 Columbia Pike

Atlanta, Georgia 30327
500 Interstate North

Dallas, Texas 75235
Suite 224 Garden Mall
Frito-Lay Building

Frankfurt, Germany
Bockenheimer Landstrasse 10
6 Frankfurt/Main

Los Angeles, California 90045
5630 Arbor Vitae Street

Miami, Florida 33131
174 East Flagler

Minneapolis, Minnesota 55408
3255 Hennepin Avenue South

Rockville, Maryland 20852
11428 Rockville Pike

St. Louis, Missouri 63108
Des Peres Hall
3694 W Pine Street

Southfield, Michigan 48075
23775 Northwestern Highway

Waltham, Massachusetts 02154
60 Hickory Drive

New York, New York 10011
Control Data Computer Training School
66 West 12 Street

San Francisco, California 94102
760 Market Street, Suite 600

Oakland, California 94612
308 16th Street

Chicago, Illinois 60603
37 South Wabash

CONTROL DATA INSTITUTE
AN EDUCATIONAL DIVISION OF
CONTROL DATA CORPORATION